

Linux Essentials

Versión 1.6
Español

010

Table of Contents



Tema 1: La Comunidad Linux y una carrera en el mundo del código abierto



1.1 Los sistemas operativos populares y la evolución de Linux

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 1.1

Importancia

2

Áreas de conocimiento clave

- Distribuciones
- Sistemas embebidos
- Linux en la nube

Lista parcial de archivos, términos y utilidades

- Debian, Ubuntu (LTS)
- CentOS, openSUSE, Red Hat, SUSE
- Linux Mint, Scientific Linux
- Raspberry Pi, Raspbian
- Android



1.1 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	1 La Comunidad Linux y una carrera en el mundo del código abierto
Objetivo:	1.1 Los sistemas operativos populares y la evolución de Linux
Lección:	1 de 1

Introducción

Linux es uno de los sistemas operativos más populares; su desarrollo se inició en 1991 por Linus Torvalds. El sistema operativo se inspiró en Unix, otro sistema operativo desarrollado en la década de 1970 por AT&T Laboratories. Unix estaba orientado a pequeñas computadoras; en ese momento, las computadoras "pequeñas" se consideraban máquinas que no necesitaban una sala completa con aire acondicionado y costaban menos de un millón de dólares. Más tarde, fueron consideradas como máquinas que podían ser transportadas por dos personas. En ese momento, Unix no estaba disponible en computadoras pequeñas como las computadoras de oficina basadas en la plataforma x86. Por lo tanto, Linus, que era un estudiante en ese momento, comenzó a implementar un sistema operativo tipo Unix que se suponía que debía ejecutarse en esta plataforma.

Principalmente, Linux usa los mismos principios e ideas básicas de Unix, pero Linux en sí no contiene código Unix, ya que es un proyecto independiente. Linux no está respaldado por una compañía individual sino por una comunidad internacional de programadores. Como está disponible gratuitamente, puede ser utilizado por cualquier persona sin restricciones.

Distribuciones

Una *distribución* de Linux es un paquete que consiste en un *kernel* de Linux y una selección de aplicaciones mantenidas por una empresa o comunidad de usuarios. El objetivo de una distribución es optimizar el núcleo y las aplicaciones que se ejecutan en el sistema operativo para un determinado caso de uso o grupo de usuarios. Las distribuciones a menudo incluyen herramientas específicas de distribución para la instalación de software y la administración del sistema. Es por esto que algunas distribuciones se usan principalmente para entornos de escritorio los cuales deben ser fáciles de usar, mientras que otras se usan principalmente para ejecutarse en servidores utilizando los recursos disponibles de la manera más eficiente posible.

Otra forma de clasificar las distribuciones es haciendo referencia a la *distribución familiar* a la que pertenecen. Las distribuciones de la familia de Debian utilizan el gestor de paquetes `dpkg` para gestionar el software que se ejecuta en el sistema operativo. Los paquetes que pueden instalarse con su gestor son mantenidos por miembros voluntarios de la comunidad. Los mantenedores utilizan el formato de paquete `deb` para especificar cómo se instala el software en el sistema operativo y cómo está configurado de forma predeterminada. Al igual que una distribución, un paquete es un conjunto de software con su correspondiente configuración y documentación que facilita el proceso de instalación, actualización y uso del software.

La distribución *Debian GNU/Linux* es la distribución más grande de la familia Debian. El proyecto Debian GNU/Linux fue lanzado por Ian Murdock en 1993 y hoy en día miles de voluntarios están trabajando en el proyecto con el objetivo de proporcionar un sistema operativo muy fiable y promover la visión de Richard Stallman de un sistema operativo que respete las libertades del usuario para ejecutar, estudiar, distribuir y mejorar el software. Esta es la razón por la cual no proporciona ningún software propietario por defecto.

Ubuntu es otra distribución basada en Debian que vale la pena mencionar. Ubuntu fue creado por Mark Shuttleworth y su equipo en 2004, con la misión de brindar un entorno de escritorio fácil de usar. La misión de Ubuntu es proporcionar software gratuito a todos en todo el mundo, así como reducir el costo de los servicios profesionales. La distribución tiene lanzamientos programados cada seis meses con un soporte a largo plazo cada 2 años.

Red Hat es una distribución de Linux desarrollada y mantenida por la compañía de software con el mismo nombre, que fue adquirida por IBM en 2019. La distribución de Red Hat Linux se inició en 1994 y se renombró en 2003 a *Red Hat Enterprise Linux*, a menudo abreviado como RHEL. Se proporciona a las empresas como una solución empresarial confiable que es compatible con Red Hat e incluye software que tiene como objetivo facilitar el uso de Linux en entornos de servidores profesionales. Algunos de sus componentes requieren suscripciones o licencias de pago. El proyecto *CentOS* utiliza el código fuente disponible de Red Hat Enterprise Linux y lo compila en una distribución que está disponible de forma totalmente gratuita, sin embargo, esta distribución no tiene

soporte comercial.

Tanto RHEL como CentOS están optimizados para su uso en entornos de servidor. El proyecto *Fedora* se fundó en 2003 y crea una distribución de Linux dirigida a computadoras de escritorio. Red Hat inició y mantiene la distribución de Fedora desde entonces. Fedora es muy progresista y adopta nuevas tecnologías muy rápidamente y a veces se considera un banco de pruebas para nuevas tecnologías que luego podrían incluirse en RHEL. Todas las distribuciones basadas en Red Hat usan el formato de paquete `rpm`.

La empresa SUSE fue fundada en 1992 en Alemania como un proveedor de servicios Unix. La primera versión de *SUSE Linux* se lanzó en 1994. A lo largo de los años, SUSE Linux se hizo conocido principalmente por su herramienta de configuración YaST. Esta herramienta permite a los administradores instalar y configurar software y hardware, así como servicios y redes. Al igual que RHEL, SUSE lanza *SUSE Linux Enterprise Server*, que es su edición comercial. Esta distribución se publica con menos frecuencia y es adecuada para la implementación empresarial y de producción. Se distribuye como un servidor, así como un entorno de escritorio con paquetes adecuados para el propósito. En 2004, SUSE lanzó el proyecto *openSUSE*, que abrió oportunidades para que los desarrolladores y usuarios probaran y desarrollaran aún más el sistema. La distribución de openSUSE está disponible gratuitamente para su descarga.

A lo largo de los años se han lanzado distribuciones independientes, algunas basadas en Red Hat o Ubuntu, otras diseñadas para mejorar una propiedad específica de un sistema o hardware. Otras construidas con funcionalidades específicas como *QubesOS*, un entorno de escritorio muy seguro, o *Kali Linux*, que proporciona un entorno para explotar las vulnerabilidades de software, utilizado principalmente por los expertos en pruebas de penetración, y otras superpequeñas distribuciones de Linux diseñadas para ejecutarse de forma específica en contenedores Linux, como Docker. También hay distribuciones creadas específicamente para componentes de sistemas embebidos e incluso dispositivos inteligentes.

Sistemas embebidos

Los sistemas embebidos son una combinación de hardware y software diseñados para tener una función específica dentro de un gran sistema. Por lo general, forman parte de otros dispositivos que ayudan a controlarlos. Los sistemas embebidos se encuentran en aplicaciones de automoción, médicas e incluso militares. Debido a su gran variedad de aplicaciones varios sistemas operativos basados en el kernel de Linux han sido desarrollados para ser utilizados en sistemas embebidos. Una parte importante de los dispositivos inteligentes tiene un sistema operativo basado en el kernel de Linux.

Por lo tanto, con los sistemas embebidos surge el software embebido. El propósito de este software es acceder al hardware y hacerlo utilizable. Entre las principales ventajas de Linux sobre cualquier

software embebido propietario se encuentran, la compatibilidad de plataformas entre vendedores, el desarrollo, el soporte y la ausencia de cuotas por concepto de licencia. Dos de los proyectos de software embebido más populares son Android, que se utiliza principalmente en teléfonos móviles a través de una variedad de proveedores, y Raspbian que se utiliza principalmente en Raspberry Pi.

Android

Android es un sistema operativo para dispositivos móviles desarrollado principalmente por Google. Android Inc. fue fundado en 2003 en Palo Alto, California. La compañía inicialmente creó un sistema operativo destinado a funcionar en cámaras digitales. En 2005, Google adquirió Android Inc. y lo desarrolló para convertirse en uno de los mayores sistemas operativos para dispositivos móviles.

La base de Android es una versión modificada del kernel de Linux con software adicional de código abierto. El sistema operativo está desarrollado principalmente para dispositivos con pantalla táctil, pero Google ha desarrollado versiones para televisores y relojes de pulsera. Se han desarrollado diferentes versiones de Android para consolas de juegos, cámaras digitales y PCs.

El código abierto de Android está disponible gratuitamente como *Android Open Source Project* (AOSP). Google ofrece una serie de componentes propietarios además del núcleo de Android. Entre los componentes se incluyen aplicaciones como Google Calendar, Google Maps, Google Mail, el navegador Chrome y Google Play Store, que facilita la instalación de aplicaciones. La mayoría de los usuarios consideran estas herramientas una parte integral de su experiencia con Android. Por lo tanto, casi todos los dispositivos móviles enviados con Android a Europa y América incluyen el software patentado de Google.

Android en dispositivos integrados tiene muchas ventajas. El sistema operativo es intuitivo y fácil de usar con una interfaz gráfica de usuario, tiene una comunidad de desarrolladores muy amplia, así que es fácil encontrar ayuda para el desarrollo. También es compatible con la mayoría de los proveedores de hardware con un controlador de Android, por lo tanto, es fácil y rentable crear un prototipo para un sistema entero.

Raspbian y Raspberry Pi

Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito que puede funcionar como un ordenador de sobremesa con todas sus funciones, pero que a su vez puede utilizarse dentro de un sistema Linux integrado. Este ordenador fue desarrollado por la Fundación Raspberry Pi, que es una organización benéfica educativa con sede en el Reino Unido. Su principal propósito es enseñar a los jóvenes a aprender a programar y comprender la funcionalidad de las computadoras. El Raspberry Pi se puede diseñar y programar para realizar tareas u operaciones que forman parte de un sistema mucho más complejo.

Las especialidades de Raspberry Pi incluyen un conjunto de pines de Entrada/Salida de Propósito

General (GPIO) que pueden ser utilizados para acoplar dispositivos electrónicos y placas de extensión, lo que permite utilizar Raspberry Pi como plataforma para el desarrollo de hardware, ya que a pesar de que estaba pensado para fines educativos se utiliza hoy en día en varios proyectos de *DIY* (hágalo usted mismo), así como para la creación de prototipos industriales en el desarrollo de sistemas embebidos.

La Raspberry Pi utiliza procesadores ARM. Varios sistemas operativos, incluido Linux, corren sobre Raspberry Pi. Como Raspberry Pi no contiene un disco duro, el sistema operativo se inicia desde una tarjeta de memoria SD. Una de las distribuciones de Linux más destacadas para Raspberry Pi es *Raspbian*. Como su nombre indica, pertenece a la familia de distribución de Debian. Está personalizado para instalarse en el hardware de Raspberry Pi y proporciona más de 35000 paquetes optimizados para este entorno. Además de Raspbian, existen muchas otras distribuciones de Linux para Raspberry Pi, como, por ejemplo, Kodi, que convierte a Raspberry Pi en un centro de medios.

Linux y el Cloud Computing

El término *cloud computing* se refiere a una forma estandarizada de consumir recursos informáticos, ya sea comprándolos a un proveedor público de cloud computing o ejecutando una nube privada. Según informes del 2017, Linux ejecuta el 90% de la carga de trabajo de la nube pública. Todos los proveedores de cloud computing, desde *Amazon Web Services* (AWS) hasta *Google Cloud Platform* (GCP) ofrecen diferentes formas de trabajar con Linux. Incluso Microsoft, una empresa cuyo antiguo CEO comparó Linux con un cáncer, ofrece hoy en día máquinas virtuales basadas en Linux en su nube *Azure*.

Linux generalmente es ofrecido como parte de *Infrastructure as a Service* (IaaS). Las instancias IaaS son máquinas virtuales que se aprovisionan en cuestión de minutos en la nube. Cuando se inicia una instancia IaaS, se elige una imagen que contiene los datos que se desplegarán en la nueva instancia. Los proveedores de nube ofrecen varias imágenes que contienen instalaciones listas para ejecutar tanto de las distribuciones populares, así como sus propias versiones de Linux. El usuario podrá elegir una imagen que contiene su distribución preferida y acceder a una instancia de la nube que ejecute esta distribución poco después de haberse creado. La mayoría de los proveedores agregan herramientas a sus imágenes para ajustar la instalación a una instancia específica de la nube. Estas herramientas pueden, por ejemplo, extender los sistemas de archivos de la imagen para que se ajusten al disco duro real de la máquina virtual.

Ejercicios guiados

1. ¿En qué se diferencia Debian GNU/Linux de Ubuntu? Nombre dos aspectos.

2. ¿Cuáles son los entornos/plataformas más comunes para los que se utiliza Linux? Nombre tres entornos/plataformas diferentes y nombre una distribución que pueda utilizarse para cada uno.

3. Se planea instalar una distribución de Linux en un nuevo entorno. Nombre cuatro aspectos que deban ser considerados al elegir una distribución.

4. Nombre tres dispositivos en los que se pueda ejecutar el sistema operativo Android, que no sean teléfonos inteligentes.

5. Explique tres ventajas importantes de la computación en la nube.

Ejercicios exploratorios

1. Teniendo en cuenta el costo y el rendimiento, ¿Qué distribuciones son más adecuadas para una empresa que tiene como objetivo reducir los costos de licencias, manteniendo el rendimiento al máximo? Explique por qué.

2. ¿Cuáles son las principales ventajas de Raspberry Pi y qué funciones pueden tener en los negocios?

3. ¿Qué gama de distribuciones ofrecen Amazon Cloud Services y Google Cloud? Nombre al menos tres comunes y dos diferentes.

Resumen

En esta lección usted aprendió:

- Diferentes distribuciones de Linux.
- ¿Qué son los sistemas embebidos Linux?
- ¿Cómo se usan los sistemas embebidos de Linux?
- Diferentes usos de Android.
- Diferentes usos de Raspberry Pi.
- ¿Qué es la computación en la nube'
- ¿Qué papel juega Linux en la computación en la nube?

Respuestas a los ejercicios guiados

1. ¿En qué se diferencia Debian GNU/Linux de Ubuntu? Nombre dos aspectos.

   Ubuntu está basado en un snapshot de Debian, es por esto que existen muchas características similares entre ellos. Sin embargo, existen otras diferencias significativas entre las dos distribuciones. La primera sería la aplicabilidad para los principiantes. Ubuntu se recomienda para principiantes por su facilidad de uso. Por otro lado, Debian se recomienda para usuarios más avanzados. La mayor diferencia está en la complejidad del proceso de instalación el cual Ubuntu hace más simple al usuario.

   Otra diferencia pudiera estar relacionada con la estabilidad de cada distribución. Debian se considera más estable comparada con Ubuntu. Debian recibe pocas actualizaciones que son probadas minuciosamente por lo que el sistema, de forma general, es más estable. Por otro lado, Ubuntu permite al usuario usar las últimas versiones de software y nuevas tecnologías.

2. ¿Cuáles son los entornos/plataformas más comunes para los que se utiliza Linux? Nombre tres entornos/plataformas diferentes y nombre una distribución que pueda utilizarse para cada uno.

   Algunos de los entornos/plataformas comunes serían teléfonos inteligentes, computadoras de escritorio y servidores. En teléfonos inteligentes, puede ser utilizado por distribuciones como Android. En escritorios y servidores, puede utilizarse por cualquier distribución, que sea adecuada a la funcionalidad de ese equipo, desde Debian, Ubuntu a CentOS y Red Hat Enterprise Linux.

3. Se planea instalar una distribución de Linux en un nuevo entorno. Nombre cuatro aspectos que deban ser considerados al elegir una distribución.

   Al elegir una distribución, algunos de los aspectos principales que se deben considerar son el costo, el rendimiento, la escalabilidad, la estabilidad y la demanda de hardware del sistema.

4. Nombre tres dispositivos en los que se pueda ejecutar el sistema operativo Android, que no sean teléfonos inteligentes.

   Otros dispositivos en los que se ejecuta Android son televisores inteligentes, tabletas, Android Auto y relojes inteligentes.

5. Explique tres ventajas importantes de la computación en la nube.

   Las principales ventajas de la computación en la nube son la flexibilidad, la fácil recuperación y el bajo costo de uso. Los servicios basados en la nube son fáciles de implementar y escalar, dependiendo de los requisitos del negocio. Tiene una gran ventaja en las soluciones de respaldo

y recuperación, ya que permite a las empresas recuperarse de los incidentes más rápido y con menos repercusiones. Además, reduce los costos de operación, ya que permite pagar solo por los recursos que utiliza una empresa en un modelo basado en suscripción.

Respuestas a los ejercicios exploratorios

1. Teniendo en cuenta el costo y el rendimiento, ¿Qué distribuciones son más adecuadas para una empresa que tiene como objetivo reducir los costos de licencias, manteniendo el rendimiento al máximo? Explique por qué.

   Una de las distribuciones más adecuadas para ser utilizada por empresas es CentOS. Esto se debe a que incorpora todos los productos de Red Hat que utiliza en su sistema operativo comercial, a la vez que es de uso gratuito. Del mismo modo, las versiones de Ubuntu LTS garantizan la compatibilidad durante un período de tiempo más largo. Las versiones estables de Debian GNU/Linux también se usan a menudo en entornos empresariales.

2. ¿Cuáles son las principales ventajas de Raspberry Pi y qué funciones pueden tener en los negocios?

   A pesar de que el RaspberryPi es muy pequeño, puede utilizarse como una computadora normal. Además, es de bajo costo y puede manejar el tráfico web y muchas otras funcionalidades. Se puede usar como un servidor, cortafuegos y se puede usar como placa principal para robots y muchos otros dispositivos pequeños.

3. ¿Qué gama de distribuciones ofrecen Amazon Cloud Services y Google Cloud? Nombre al menos tres comunes y dos diferentes.

   Las distribuciones comunes entre Amazon y Google Cloud Services son Ubuntu, CentOS y Red Hat Enterprise Linux. Cada proveedor de la nube también ofrece distribuciones específicas que el otro no ofrece. Amazon tiene Amazon Linux y Kali Linux, mientras que Google ofrece el uso de servidores FreeBSD y Windows.



1.2 Principales aplicaciones de código abierto

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 1.2

Importancia

2

Áreas de conocimiento clave

- Aplicaciones de escritorio
- Aplicaciones de servidor
- Lenguajes de programación
- Herramientas de gestión de paquetes y repositorios

Lista parcial de archivos, términos y utilidades

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Nextcloud, ownCloud
- Apache HTTPD, NGINX, MariaDB, MySQL, NFS, Samba
- C, Java, JavaScript, Perl, shell, Python, PHP
- dpkg, apt-get, rpm, yum



1.2 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	1 La comunidad Linux y una carrera en el Open Source
Objetivo:	1.2 Principales aplicaciones de código abierto
Lección:	1 de 1

Introducción

Una aplicación es un programa informático que no está directamente relacionado con el funcionamiento de la computadora, sino de las tareas que realiza el usuario. Las distribuciones de Linux ofrecen muchas opciones de aplicaciones para realizar una variedad de tareas, tales como aplicaciones de oficina, navegadores web, reproductores y editores multimedia, etc. A menudo hay más de una aplicación o herramienta para realizar un trabajo en particular, y es el usuario quien debe elegir la aplicación que mejor se adapte a sus necesidades.

Paquetes de software

Casi todas las distribuciones de Linux ofrecen un conjunto aplicaciones preinstaladas. Además de esas aplicaciones preinstaladas, una distribución tiene un repositorio de paquetes con una vasta colección de aplicaciones disponibles para instalar a través de su gestor de paquetes *package manager*. Aunque las diversas distribuciones ofrecen aproximadamente las mismas aplicaciones, existen varios sistemas de gestión de paquetes diferentes para varias distribuciones, por ejemplo, Debian, Ubuntu y Linux Mint utilizan las herramientas `dpkg`, `apt-get` y `apt` para instalar paquetes de software, generalmente denominados paquetes *DEB*. Por el contrario, distribuciones como Red Hat,

Fedora y CentOS usan los comandos `rpm`, `yum` y `dnf` que a su vez instalan paquetes *RPM*. Como el empaquetado de la aplicación es diferente para cada familia de distribución, es muy importante instalar paquetes desde el repositorio correcto diseñado para la distribución en particular. El usuario final normalmente no tiene que preocuparse por esos detalles, ya que el gestor de paquetes de la distribución elegirá los paquetes adecuados, las dependencias necesarias y las actualizaciones futuras. Las dependencias son paquetes auxiliares que necesitan los programas. Por ejemplo, si una librería proporciona funciones para tratar imágenes JPEG que son utilizadas por varios programas, es probable que esta librería esté empaquetada en su propio paquete del que dependen todas las aplicaciones que utilizan la librería.

Los comandos `dpkg` y `rpm` operan en archivos de paquetes individuales. En la práctica, casi todas las tareas de gestión de paquetes son realizadas por los comandos `apt-get` o `apt` en sistemas que utilizan paquetes DEB o por `yum` o `dnf` en sistemas que utilizan paquetes RPM. Estos comandos funcionan con catálogos de paquetes. Se puede descargar nuevos paquetes y sus dependencias, además comprobar si hay versiones más nuevas de los paquetes instalados.

Instalación de paquetes

Suponga que ha oído hablar de un comando llamado `figlet` que imprime texto agrandado en el terminal y desea probarlo; sin embargo, obtendrá el siguiente mensaje después de ejecutar el comando `figlet`:

```
$ figlet
-bash: figlet: command not found
```

Esto significa que probablemente el paquete no esté instalado en su sistema. Si su distribución funciona con paquetes DEB, puede buscar en sus repositorios usando `apt-cache search package_name` o `apt search package_name`. El comando `apt-cache` se usa para buscar paquetes y para listar información sobre paquetes disponibles. El siguiente comando busca cualquier ocurrencia del término “figlet” en los nombres y descripciones del paquete:

```
$ apt-cache search figlet
figlet - Make large character ASCII banners out of ordinary text
```

La búsqueda identificó un paquete llamado *figlet* que corresponde al comando que falta. La instalación y eliminación de un paquete requiere permisos especiales otorgados solo al administrador del sistema: el usuario llamado `root`. En los sistemas de escritorio, los usuarios comunes pueden instalar o eliminar paquetes anteponiendo el comando `sudo` a los comandos de instalación / eliminación. Eso requerirá que escriba su contraseña para continuar. Para los paquetes DEB, la

instalación se realiza con el comando `apt-get install package_name` o `apt install package_name`:

```
$ sudo apt-get install figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

En este punto el paquete se descargará e instalará en el sistema, así como las dependencias que eventualmente necesite el paquete:

```
Need to get 184 kB of archives.
After this operation, 741 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian stretch/main armhf figlet armhf 2.2.5-2
[184 kB]
Fetched 184 kB in 0s (213 kB/s)
Selecting previously unselected package figlet.
(Reading database ... 115701 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_armhf.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet
(figlet) in auto mode
Processing triggers for man-db (2.7.6.1-2) ...
```

Una vez finalizada la descarga, todos los archivos son copiados a las ubicaciones correspondiente, se realizará cualquier configuración adicional y el comando estará disponible:

```
$ figlet Awesome!
    _                                         _
   / \__      _____  ___  ___  _ __ ___   ___| |
  / _ \ \ /\ / / _ \/ __|/ _ \| '_ ` _ \ / _ \ |
 / ___ \ V  V /  __/\__ \ (_) | | | | | |  __/_|
/_/   \_\_/\_/ \___||___/\___/|_| |_| |_|\___(_)
```

En distribuciones basadas en paquetes RPM, las búsquedas se realizan usando `yum search package_name` o `dnf search package_name`. Supongamos que desea mostrar algún texto de una forma más irreverente, seguido de una vaca caricaturesca, pero no está seguro del paquete que puede

realizar esa tarea:

```
$ yum search speaking cow
Last metadata expiration check: 1:30:49 ago on Tue 23 Apr 2019 11:02:33 PM -03.
==================== Name & Summary Matched: speaking, cow ====================
cowsay.noarch : Configurable speaking/thinking cow
```

Después de encontrar un paquete adecuado en el repositorio, puede instalarse con `yum install package_name` o `dnf install package_name`:

```
$ sudo yum install cowsay
Last metadata expiration check: 2:41:02 ago on Tue 23 Apr 2019 11:02:33 PM -03.
Dependencies resolved.
===============================================================================
 Package         Arch           Version                Repository         Size
===============================================================================
Installing:
 cowsay          noarch         3.04-10.fc28           fedora             46 k

Transaction Summary
===============================================================================
Install  1 Package

Total download size: 46 k
Installed size: 76 k
Is this ok [y/N]: y
```

Una vez más, el paquete deseado y todas sus posibles dependencias serán descargados e instalados:

```
Downloading Packages:
cowsay-3.04-10.fc28.noarch.rpm                    490 kB/s |  46 kB     00:00
===============================================================================
Total                                              53 kB/s |  46 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing        :                                                        1/1
Installing       : cowsay-3.04-10.fc28.noarch                             1/1
Running scriptlet: cowsay-3.04-10.fc28.noarch                             1/1
Verifying        : cowsay-3.04-10.fc28.noarch                             1/1

Installed:
cowsay.noarch 3.04-10.fc28

Complete!
```

El comando `cowsay` hace exactamente lo que su nombre indica:

```
$ cowsay "Brought to you by yum"
 _______________________
< Brought to you by yum >
 -----------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Aunque puedan parecer inútiles, los comandos `figlet` y `cowsay` proporcionan una forma de llamar la atención de otros usuarios sobre la información relevante.

Eliminación de paquetes

Los mismos comandos utilizados para instalar paquetes se utilizan para eliminarlos. Todos los comandos aceptan la palabra clave `remove` para desinstalar un paquete instalado: `apt-get remove package_name` o `apt remove package_name` para paquetes DEB y `yum remove package_name` o `dnf remove package_name` para paquetes RPM. El comando `sudo` también es necesario para realizar la eliminación. Por ejemplo, para eliminar el paquete *figlet* instalado previamente en una distribución

basada en DEB:

```
$ sudo apt-get remove figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  figlet
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 741 kB disk space will be freed.
Do you want to continue? [Y/n] Y
```

Después de confirmar la operación, el paquete se borra del sistema:

```
(Reading database ... 115775 files and directories currently installed.)
Removing figlet (2.2.5-2) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Un procedimiento similar se realiza en un sistema basado en RPM, por ejemplo, para eliminar el paquete *cowsay* previamente instalado de una distribución basada en RPM:

```
$ sudo yum remove cowsay
Dependencies resolved.
==================================================================================
 Package          Arch              Version                Repository        Size
==================================================================================
Removing:
 cowsay           noarch            3.04-10.fc28           @fedora           76 k

Transaction Summary
==================================================================================
Remove  1 Package

Freed space: 76 k
Is this ok [y/N]: y
```

Asimismo, se solicita una confirmación y se borra el paquete del sistema:

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Erasing          : cowsay-3.04-10.fc28.noarch                             1/1
  Running scriptlet: cowsay-3.04-10.fc28.noarch                             1/1
  Verifying        : cowsay-3.04-10.fc28.noarch                             1/1

Removed:
  cowsay.noarch 3.04-10.fc28

Complete!
```

Los archivos de configuración de los paquetes eliminados se mantienen en el sistema, por lo que pueden volver a utilizarse si el paquete se vuelve a instalar en el futuro.

Aplicaciones de Office

Las aplicaciones de Office se utilizan para editar archivos como textos, presentaciones, hojas de cálculo y otros formatos que se utilizan habitualmente en un entorno de oficina, y que suelen organizarse en colecciones denominadas *office suites*.

Durante mucho tiempo, la suite de oficina más utilizada en Linux fue la suite *OpenOffice.org*. OpenOffice.org era una versión de código abierto de la suite *StarOffice* lanzada por *Sun Microsystems*. Unos años más tarde, Sun fue adquirida por *Oracle Corporation*, que a su vez transfirió el proyecto a la *Fundación Apache* y OpenOffice.org pasó a llamarse *Apache OpenOffice*. Mientras tanto, otra suite de oficina basada en el mismo código fuente fue lanzada por la *Document Foundation*, que la denominó *LibreOffice*.

Los dos proyectos tienen las mismas características básicas y son compatibles con los formatos de documentos de *Microsoft Office*. Sin embargo, el formato de documento preferido es *Open Document Format*, un formato de archivo completamente abierto y estandarizado ISO. El uso de archivos ODF garantiza que los documentos se puedan transferir entre sistemas operativos y aplicaciones de diferentes proveedores, como Microsoft Office. Las principales aplicaciones que ofrece OpenOffice/LibreOffice son:

Writer
: Editor de texto

Calc
: Hojas de cálculo

Impress
: Presentaciones

Draw
: Dibujo vectorial

Math
: Fórmulas matemáticas

Base
: Base de datos

Tanto LibreOffice como Apache OpenOffice son software de código abierto, pero LibreOffice está licenciado bajo LGPLv3 y Apache OpenOffice está licenciado bajo Apache License 2.0. La distinción de licencia implica que LibreOffice puede incorporar mejoras hechas por Apache OpenOffice, pero Apache OpenOffice no puede incorporar mejoras hechas por LibreOffice. Por esa razón y por una comunidad más activa de desarrolladores, las distribuciones más populares que adoptan LibreOffice como su suite ofimática predeterminada.

Navegadores web

Para la mayoría de los usuarios, el objetivo principal de una computadora es proporcionar acceso a Internet. Hoy en día, las páginas web pueden actuar como una aplicación completa, con la ventaja de ser accesibles desde cualquier lugar, sin la necesidad de instalar software adicional, lo que hace que el navegador web sea la aplicación más importante del sistema operativo, al menos para el usuario medio.

> **TIP** Una de las mejores fuentes para aprender sobre el desarrollo web es MDN Web Docs, disponible en https://developer.mozilla.org/. El sitio es mantenido por Mozilla y está lleno de tutoriales para principiantes y materiales de referencia sobre la mayoría de las tecnologías web modernas.

Los principales navegadores web en el entorno Linux son *Google Chrome* y *Mozilla Firefox*. Chrome es un navegador web mantenido por Google pero se basa en el navegador de código abierto llamado *Chromium*, que puede instalarse utilizando el gestor de paquetes de la distribución y es totalmente compatible con Chrome. Mantenido por Mozilla, una organización sin fines de lucro, Firefox es un navegador cuyos orígenes están ligados a Netscape, el primer navegador web popular en adoptar el

modelo de código abierto, y que está profundamente involucrado con el desarrollo de los estándares abiertos que subyacen en la web moderna.

Mozilla también desarrolla otras aplicaciones, como el cliente de correo electrónico *Thunderbird*. Muchos usuarios optan por utilizar webmail en lugar de una aplicación de correo electrónico dedicada, pero un cliente como Thunderbird ofrece funciones adicionales y se integra mejor con otras aplicaciones en el escritorio.

Multimedia

En comparación con las aplicaciones web disponibles, las aplicaciones de escritorio siguen siendo la mejor opción para la creación de contenido multimedia. Las actividades como la renderización de video, a menudo requieren altos niveles en los recursos del sistema que son mejor administrados por una aplicación de escritorio local. Algunas de las aplicaciones multimedia más populares para Linux y sus usos se enumeran a continuación.

Blender
: Un renderizador 3D para crear animaciones, también se puede utilizar para exportar objetos 3D para ser impresos por una impresora 3D.

GIMP
: Un editor de imágenes completo, que puede compararse con *Adobe Photoshop*, pero que tiene sus propios conceptos y herramientas para trabajar con imágenes. GIMP puede utilizarse para crear, editar y guardar la mayoría de los archivos de mapas de bits, como JPEG, PNG, GIF, TIFF y muchos otros.

Inkscape
: Un editor de gráficos vectoriales, similar a *Corel Draw* o *Adobe Illustrator*. El formato por defecto de Inkscape es SVG, que es un estándar abierto para gráficos vectoriales, los archivos SVG pueden ser abiertos por cualquier navegador web y, debido a su naturaleza de gráfico vectorial, puede ser utilizado en diseños flexibles de páginas web.

Audacity
: Un editor de audio que permite filtrar, aplicar efectos y convertir a diferentes formatos de audio, como MP3, WAV, OGG, FLAC, etc.

ImageMagick
: ImageMagick es una herramienta de línea de comandos para convertir y editar la mayoría de los archivos de tipo imagen. También puede ser usado para crear documentos PDF desde un archivo de imagen y viceversa.

También hay muchas aplicaciones dedicadas a la reproducción multimedia. La aplicación más popular para la reproducción de video es *VLC*, pero algunos usuarios prefieren otras alternativas, como *smplayer*. La reproducción de música local también tiene muchas opciones, como *Audacious*, *Banshee* y *Amarok*, que también pueden administrar una colección de archivos de sonido locales.

Programas para servidores

Cuando un navegador web carga una página de un sitio, en realidad se está conectando a un ordenador remoto y solicitando una información específica. En este escenario, el ordenador que ejecuta el navegador web se llama *cliente* y el ordenador remoto se llama *servidor*.

El servidor, puede ser un ordenador de sobremesa ordinario o un hardware especializado, este equipo necesita un programa o servicio (Conocido también de esa forma) específico para gestionar cada tipo de información que proporcione. En lo que se refiere al servicio de páginas web, la mayoría de los servidores de todo el mundo utilizan programas de código abierto. A este programa o servicio en particular se le llama servidor *HTTP* (HTTP significa *Protocolo de Transferencia de Texto Hyper*) y los más populares son *Apache*, *Nginx* y *lighttpd*.

Incluso las páginas web simples pueden requerir muchas solicitudes, que podrían ser archivos ordinarios como --contenido estático-- o --contenido dinámico-- generado desde varias fuentes. La función de un servidor HTTP es recopilar y enviar todos los datos solicitados al navegador, que a su vez organiza el contenido según lo definido por el documento HTML (*Hyper Text Markup Language*) recibido por el servidor, en conjunto a otros tipos de archivos que soporta el navegador, por lo que el renderizado de una página web implica operaciones realizadas en el lado del servidor y del cliente. Ambas partes pueden utilizar scripts personalizados para realizar tareas específicas. En el lado del servidor HTTP, es bastante común usar el lenguaje programación PHP. En el lado del cliente, JavaScript es el lenguaje de programación utilizado en el navegador web.

Los programas de servidores pueden proporcionar todo tipo de información, no es raro tener un programa de servidor que solicite información proporcionada por otros servicios, como es el caso cuando un servidor HTTP requiere información proporcionada por un servidor de base de datos.

Por ejemplo, cuando se solicita una página dinámica, el servidor HTTP suele consultar una base de datos para recoger toda la información necesaria y enviar el contenido dinámico de vuelta al cliente, del mismo modo que cuando un usuario se registra en un sitio web, el servidor HTTP recoge los datos enviados por el cliente y los almacena en una base de datos.

Una base de datos es un conjunto organizado de información. Un servidor de base de datos almacena contenido de manera formateada, lo que permite leer, escribir y vincular grandes cantidades de datos de manera confiable y a gran velocidad. Los servidores de bases de datos de código abierto se utilizan en muchas aplicaciones, no solo en Internet. Incluso las aplicaciones locales pueden almacenar datos

conectándose a un servidor de base de datos local. El tipo más común de base de datos es la *base de datos relacional*, donde los datos se organizan en tablas predefinidas. Las bases de datos relacionales de código abierto más populares son *MariaDB* (originado en *MySQL*) y *PostgreSQL*.

Datos compartidos (Data Sharing)

En las redes locales, como las que se encuentran en las oficinas y los hogares, es deseable que los ordenadores no sólo puedan acceder a Internet, sino que también puedan comunicarse entre sí. En ocasiones un ordenador puede actuar como un servidor y un cliente. Por ejemplo, cuando se desea acceder a los archivos de otro ordenador de la red (Supongamos, que se necesita acceder a un archivo almacenado en un ordenador de sobremesa desde un dispositivo portátil) sin la molestia de tener que copiarlo en una unidad USB o similar.

Entre máquinas Linux, *NFS* (*Network File System*) se utiliza a menudo para este tipo de funciones. El protocolo NFS es la forma estándar de compartir sistemas de archivos en redes equipadas únicamente con máquinas Unix/Linux con las que un ordenador puede compartir uno o varios de sus directorios con ordenadores específicos de la red. De este modo, NFS puede leer y escribir archivos en estos directorios. Incluso se puede usar NFS para compartir el árbol de directorios de todo un sistema operativo con clientes que lo usarán para arrancar (*Bootear*) desde el servidor. Estas computadoras, llamadas *thin clients*, se utilizan principalmente en redes grandes para evitar el mantenimiento de cada sistema operativo de cada máquina.

Si existen otros tipos de sistemas operativos conectados a la red, se recomienda utilizar un protocolo de datos compartidos comprensible para todos ellos. Este requisito lo cumple *Samba*. Samba implementa un protocolo de datos compartidos a través de la red. Este fue originalmente diseñado para el sistema operativo Windows, pero que hoy en día es compatible con todos los principales sistemas operativos, ya que con Samba los ordenadores de la red local no sólo pueden compartir archivos, sino también impresoras.

En algunas redes locales, la autorización dada al iniciar sesión en una estación de trabajo es otorgada por un servidor central, llamado *controlador de dominio*, conocido como *domain controller*, que gestiona el acceso a varios recursos locales y remotos. el controlador de dominio es un servicio proporcionado por el *Active Directory* de Microsoft. Las estaciones de trabajo Linux pueden asociarse con un controlador de dominio mediante el uso de Samba o un subsistema de autenticación llamado *SSSD*. A partir de la versión 4, Samba también puede funcionar como un controlador de dominio en redes heterogéneas.

Si el objetivo es implementar una solución de computación en la nube (*Cloud Computing*) capaz de proporcionar varios métodos de compartir datos basados en web; deberá considerar dos alternativas: *ownCloud* y *Nextcloud*. Los dos proyectos son muy similares porque Nextcloud es un *spin-off* de ownCloud, lo cual no es inusual entre los proyectos de código abierto; a esto se le denomina *fork*.

Ambos proporcionan las mismas características básicas: compartición y sincronización de archivos, espacios de trabajo colaborativos, calendario, contactos y correo, todo a través de interfaces de escritorio, móviles y web. Nextcloud también proporciona conferencias privadas de audio/vídeo, mientras que ownCloud se centra más en el intercambio de archivos y la integración con software de terceros. Muchas más características se proporcionan como plugins que pueden ser activados más tarde según sea necesario.

Tanto ownCloud como Nextcloud ofrecen una versión paga con características adicionales y soporte extendido. Lo que los hace diferentes de otras soluciones comerciales es la capacidad de instalar Nextcloud o ownCloud en un servidor privado, de forma gratuita, evitando mantener datos confidenciales en un servidor desconocido. Como todos los servicios dependen de la comunicación HTTP y están escritos en PHP, la instalación debe realizarse en un servidor web configurado previamente, como Apache. Si considera instalar ownCloud o Nextcloud en su propio servidor, asegúrese de habilitar también HTTPS para cifrar todas las conexiones a su nube.

Administración de la red

La comunicación entre computadoras solo es posible si la red funciona correctamente. Normalmente, la configuración de la red se realiza mediante un conjunto de programas que se ejecutan en el router, encargados de configurar y comprobar la disponibilidad de la red. Para esto se utilizan dos servicios de red básicos: *DHCP* (*Dynamic Host Configuration Protocol*) y *DNS* (*Domain Name System*).

DHCP es responsable de asignar una dirección IP al host cuando se conecta un cable de red o cuando el dispositivo entra en una red inalámbrica. Al conectarse a Internet, el servidor DHCP del proveedor de servicios de Internet proporcionará una dirección IP al dispositivo solicitante. Un servidor DHCP es muy útil en las redes de área local para proporcionar automáticamente direcciones IP a todos los dispositivos conectados. Si DHCP no está configurado o si no funciona correctamente, sería necesario configurar manualmente la dirección IP de cada dispositivo conectado a la red, lo cual no es práctico entornos de redes grandes e inclusive pequeñas.

La dirección IP es necesaria para comunicarse con otro dispositivo en una red, pero es mucho más probable que las personas recuerden los nombres de dominio como `www.lpi.org` que un número IP como `203.0.113.113.165`. Sin embargo, el nombre de dominio por sí solo no es suficiente para establecer la comunicación a través de la red, por lo que necesita ser traducido a una dirección IP por un servidor DNS. La dirección IP del servidor DNS es suministrada por el servidor DHCP del proveedor de servicios de Internet (ISP) y es utilizada por todos los sistemas conectados para traducir los nombres de dominio a direcciones IP.

Las configuraciones de DHCP y DNS se pueden modificar ingresando a la interfaz web provista por el enrutador. Por ejemplo, es posible restringir la asignación de IP sólo a dispositivos conocidos o asociar una dirección IP fija a máquinas específicas. También es posible cambiar el servidor DNS

proporcionado por el ISP por algunos servidores DNS de terceros, como los de Google o OpenDNS, en ocasiones pueden proporcionar respuestas más rápidas y funciones adicionales.

Lenguajes de programación

Todos los programas de los ordenadores (programas cliente y servidor, aplicaciones de escritorio y el propio sistema operativo) se realizan utilizando uno o más lenguajes de programación, ya sea un único archivo o un complejo sistema de cientos de archivos que el sistema operativo trata como una secuencia de instrucciones que deben ser interpretadas y ejecutadas por el procesador y otros dispositivos.

Existen numerosos lenguajes de programación para propósitos muy diferentes y los sistemas Linux proporcionan muchos de ellos. Como el software de código abierto también incluye las fuentes de los programas, los sistemas Linux ofrecen a los desarrolladores condiciones perfectas para entender, modificar o crear software de acuerdo con sus propias necesidades.

Cada programa comienza como un archivo de texto, llamado *source code*. Este código fuente está escrito en un lenguaje más o menos amigable con el ser humano que describe lo que está haciendo el programa. Un procesador no puede ejecutar directamente este código, por lo que en *lenguajes de compilación* el código fuente se convierte en un *archivo binario* (*binary file*) que puede ser ejecutado por el ordenador, mientras que un programa llamado *compilador* (*compiler*) se encarga de realizar la conversión del código fuente a una forma ejecutable. Dado que el binario compilado es específico para un tipo de procesador, es posible que el programa deba volver a compilarse para ejecutarse en otro tipo de computadora.

En *lenguajes interpretados* (*interpreted languages*), el programa no necesita ser compilado previamente, sino que un *intérprete* lee el código fuente y ejecuta su instrucción cada vez que se ejecuta el programa, lo que hace que el desarrollo sea más fácil y rápido, pero al mismo tiempo los programas interpretados tienden a ser más lentos que los compilados.

Aquí algunos de los lenguajes de programación más populares:

JavaScript

JavaScript es un lenguaje de programación muy utilizado en páginas web, en sus orígenes, las aplicaciones JavaScript eran muy sencillas, como las rutinas de validación de formularios. Hoy en día, JavaScript es considerado un lenguaje de primera clase y se utiliza para crear aplicaciones muy complejas no sólo en la web, sino también en servidores y dispositivos móviles.

C

El lenguaje de programación C está estrechamente relacionado con los sistemas operativos, en particular con Unix, pero se utiliza para escribir cualquier tipo de programa en casi cualquier tipo

de dispositivo. Las grandes ventajas de C son la flexibilidad y la velocidad; el mismo código fuente escrito en C puede ser compilado para ejecutarse en diferentes plataformas y sistemas operativos con poca o ninguna modificación, sin embargo, después de ser compilado, el programa se ejecutará sólo en el sistema donde se compiló.

Java

El aspecto principal de Java es que los programas escritos en este lenguaje son portátiles, lo que significa que el mismo programa puede ejecutarse en diferentes sistemas operativos. A pesar de su nombre, Java no está relacionado con JavaScript.

Perl

Perl es un lenguaje de programación muy utilizado para procesar contenido de texto con un fuerte énfasis en las expresiones regulares, lo que hace de Perl un lenguaje adecuado para el filtrado y análisis de texto.

Shell

El shell, en particular el Bash shell, no es sólo un lenguaje de programación, sino una interfaz interactiva para ejecutar otros programas. Los programas de Shell, conocidos como *shell scripts* pueden automatizar tareas complejas o repetitivas en el entorno de línea de comandos.

Python

Python es un lenguaje de programación muy popular entre estudiantes y profesionales que no están directamente relacionados con la informática. Aunque tiene características avanzadas, Python es una buena manera de empezar a aprender programación por su enfoque fácil de usar.

PHP

PHP es el lenguaje de programación más utilizado en el lado del servidor para generar contenido para la web, la mayoría de las páginas HTML en línea no son archivos estáticos, sino contenido dinámico generado por el servidor desde varias fuentes, como bases de datos. Los programas PHP — a veces sólo llamados páginas PHP o scripts PHP — se utilizan a menudo para generar este tipo de contenido. El término LAMP proviene de la combinación de un sistema operativo Linux, un servidor Apache HTTP, una base de datos MySQL (o MariaDB) y la programación PHP. Los servidores LAMP son una solución muy popular para la ejecución de servidores web.

C y Java son lenguajes compilados, para ser ejecutados por el sistema, el código fuente escrito en C se convierte en código binario de la máquina, mientras que el código fuente Java se convierte en *bytecode* ejecutado en un entorno de software especial llamado *Java Virtual Machine*. JavaScript, Perl, Shell script, Python y PHP son lenguajes interpretados, también llamados *lenguajes de script*.

Ejercicios guiados

1. Para cada uno de los siguientes comandos, identifique si está asociado con el sistema de empaquetado Debian (*Debian packaging system*) o con el sistema de empaquetado Red Hat (*Red Hat packaging system*) :

`dpkg`	
`rpm`	
`apt-get`	
`yum`	
`dnf`	

2. ¿Qué comando se puede usar para instalar Blender en Ubuntu? Después de la instalación, ¿Cómo se puede ejecutar el programa?

3. ¿Qué aplicación del paquete LibreOffice se puede utilizar para trabajar con hojas de cálculo?

4. ¿Qué navegador web de código abierto se utiliza como base para el desarrollo de Google Chrome?

5. SVG es un estándar abierto para gráficos vectoriales. ¿Cuál es la aplicación más popular para editar archivos SVG en sistemas Linux?

6. Para cada uno de los siguientes formatos de archivo, escriba el nombre de una aplicación capaz de abrir y editar el archivo correspondiente:

`png`	
`doc`	
`xls`	
`ppt`	
`wav`	

7. ¿Qué paquete de software permite compartir archivos entre máquinas Linux y Windows a través de la red local?

Ejercicios exploratorios

1. Usted sabe que los archivos de configuración se mantienen incluso si el paquete asociado se elimina del sistema. ¿Cómo puede eliminar automáticamente el paquete llamado *cups* y sus archivos de configuración de un sistema basado en DEB?

2. Suponga que tiene muchos archivos de imagen TIFF y desea convertirlos a JPEG. ¿Qué paquete de software podría utilizarse para convertir esos archivos directamente en la línea de comandos?

3. ¿Qué paquete de software necesita instalar para poder abrir documentos de Microsoft Word enviados por un usuario de Windows?

4. Cada año, linuxquestions.org promueve una encuesta sobre las aplicaciones Linux más populares. Visite https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/ y descubra qué aplicaciones de escritorio son las más populares entre los usuarios experimentados de Linux.

Resumen

En esta lección usted aprendió:

- Los sistemas de gestión de paquetes utilizados en las principales distribuciones de Linux.
- Aplicaciones de código abierto que pueden editar formatos de archivo populares.
- Los programas de servidor subyacentes a muchos servicios importantes de Internet y de redes locales.
- Lenguajes de programación comunes y sus usos.

Respuestas a los ejercicios guiados

1. Para cada uno de los siguientes comandos, identifique si está asociado con el sistema de empaquetado Debian (*Debian packaging system*) o con el sistema de empaquetado Red Hat (*Red Hat packaging system*) :

`dpkg`	Debian sistema de empaquetado
`rpm`	Red Hat sistema de empaquetado
`apt-get`	Debian sistema de empaquetado
`yum`	Red Hat sistema de empaquetado
`dnf`	Red Hat sistema de empaquetado

2. ¿Qué comando se puede usar para instalar Blender en Ubuntu? Después de la instalación, ¿cómo se puede ejecutar el programa?

   El comando `apt-get install blender`. El nombre del paquete debe especificarse en minúscula. El programa se puede ejecutar directamente desde el terminal con el comando `blender` o seleccionándolo en el menú de aplicaciones.

3. ¿Qué aplicación del paquete LibreOffice se puede utilizar para trabajar con hojas de cálculo electrónicas?

   Calc

4. ¿Qué navegador web de código abierto se utiliza como base para el desarrollo de Google Chrome?

   Chromium

5. SVG es un estándar abierto para gráficos vectoriales. ¿Cuál es la aplicación más popular para editar archivos SVG en sistemas Linux?

   Inkscape

6. Para cada uno de los siguientes formatos de archivo, escriba el nombre de una aplicación capaz de abrir y editar el archivo correspondiente:

`png`	Gimp
`doc`	LibreOffice Writer

`xls`	LibreOffice Calc
`ppt`	LibreOffice Impress
`wav`	Audacity

7. ¿Qué paquete de software permite compartir archivos entre máquinas Linux y Windows a través de la red local?

   Samba

Respuestas a los ejercicios exploratorios

1. Usted sabe que los archivos de configuración se mantienen incluso si el paquete asociado se elimina del sistema. ¿Cómo puede eliminar automáticamente el paquete llamado *cups* y sus archivos de configuración de un sistema basado en DEB?

   ```
   apt-get purge cups
   ```

2. Suponga que tiene muchos archivos de imagen TIFF y desea convertirlos a JPEG. ¿Qué paquete de software podría utilizarse para convertir esos archivos directamente en la línea de comandos?

   ImageMagick

3. ¿Qué paquete de software necesita instalar para poder abrir documentos de Microsoft Word enviados por un usuario de Windows?

   LibreOffice - OpenOffice

4. Cada año, linuxquestions.org promueve una encuesta sobre las aplicaciones Linux más populares. Visite https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/ y descubra qué aplicaciones de escritorio son las más populares entre los usuarios experimentados de Linux.

   Navegador web: Firefox. Cliente de correo: Thunderbird. Reproductor de medios: VLC. Editor gráfico: GIMP.



1.3 Software de Código Abierto y las licencias

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 1.3

Importancia

1

Áreas de conocimiento clave

- Filosofía del Código Abierto
- Licencias de Código Abierto
- Free Software Foundation (FSF), Open Source Initiative (OSI)

Lista parcial de archivos, términos y utilidades

- Copyleft, Permissive
- GPL, BSD, Creative Commons
- Free Software, Open Source Software, FOSS, FLOSS
- Modelos de negocios basados en el software de código abierto



1.3 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	1 La comunidad Linux y una carrera en el Open Source
Objetivo:	1.3 Software de código abierto y licenciamiento
Lección:	1 de 1

Introducción

Aunque los términos *software libre* y *software de código abierto* ("Open Source") se utilizan ampliamente, todavía existen algunos conceptos erróneos acerca de su significado. En particular el concepto "libertad" requiere de una conceptualización mucho más detallada. Así que comencemos con la definición de los dos términos.

Definición de Software Libre y de Código Abierto

Criterios de Software Libre

En primer lugar, la palabra "libre" en el contexto del software libre no tiene nada que ver con "libre de cargo" o como el fundador de la *Free Software Foundation* (FSF), Richard Stallman, lo expresa:

> Para entender el concepto, deberías pensar en "libre" como "libre expresión", no como en "cerveza libre".
>
> — Richard Stallman, ¿Qué es el software libre?

Independientemente de si hay que pagar por el software o no, hay cuatro criterios que constituyen el software libre. Richard Stallman describe estos criterios como “las cuatro libertades esenciales” cuyo recuento comienza desde cero:

- “La libertad de ejecutar el programa como se desee, para cualquier propósito (libertad 0).”

  Dónde, cómo y con qué propósito el software utilizado no puede prescribirse ni restringirse.

- “La libertad de estudiar cómo funciona el programa y de modificarlo para que se adapte a tu necesidad (libertad 1). El acceso al código fuente es una condición previa para ello.”

  Cualquier persona puede cambiar el software en función de sus ideas y necesidades, lo que a su vez da por hecho que el código fuente, es decir, todos los archivos que componen un software, estén disponibles en un formato legible para los programadores. Por supuesto, este derecho se aplica a un usuario que quiera agregar alguna característica o función al código; así como lo hacen las compañías de software que construyen sistemas complejos como los sistemas operativos de teléfonos inteligentes (smartphone) o el firmware de un enrutador (router).

- “La libertad de redistribuir copias para poder ayudar a los demás (libertad 2).”

  Esta libertad anima explícitamente a cada usuario a compartir el software con los demás, por lo que se trata de una distribución lo más amplia posible, y lo por tanto extender a una comunidad de usuarios y desarrolladores que toman la base de estas libertades para desarrollar y mejorar aún más el software en beneficio de todos.

- “La libertad de distribuir copias de sus versiones modificadas a otros (libertad 3). De esta forma, puede dar a toda la comunidad la oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición previa para ello.”

  Esto no es sólo acerca de la distribución de software libre, sino de distribuir las *modificaciones* que se realizan. Cualquier persona que realice cambios en el software libre tiene derecho a que los cambios estén disponibles para otros. Si lo hacen, también están obligados a hacerlo libremente, es decir, no deben restringir las libertades originales al distribuirlo, incluso si lo modifican o lo amplían. Por ejemplo, si un grupo de desarrolladores tiene ideas diferentes a la de los autores originales con respecto a la dirección de un software; podrían separar su propia rama de desarrollo (*fork*) y continuarla como un nuevo proyecto, pero permaneciendo todas las obligaciones asociadas con estas libertades.

El énfasis de libertad también consistente en la medida de que todo movimiento de libertad está dirigido *en contra de algo*, es decir, alguien que suprime las libertades postuladas, que considera el software como propiedad y quiere mantenerlo bajo llave, a diferencia del software libre. Dicho

software se denomina *propietario*.

Software de código abierto vs. Software libre

Para muchos, software libre y software de código abierto son sinónimos. La abreviatura *FOSS* de *Free and Open Source Software* enfatiza este punto en común, mientras que FLOSS de *Free/Libre and Open Source Software* es otro término popular que inequívocamente enfatiza la idea de libertad también para otros idiomas que no sea el inglés, sin embargo, si se considera el origen y desarrollo de ambos términos, una diferenciación vale la pena.

El término *software libre* con la definición de las cuatro libertades se remonta a Richard Stallman y el proyecto GNU fundado en 1985 por él mismo, casi 10 años antes de la aparición de Linux. El nombre "GNU no es Unix" describe la intención con un abrir y cerrar de ojos: GNU comenzó con la iniciativa de desarrollar una solución técnicamente convincente — como el sistema operativo Unix — desde cero para disposición del público en general y mejorarlo continuamente con el que quisiera involucrase. La apertura del código fuente era simplemente una necesidad técnica y organizativa, pero en su propia imagen, el movimiento del software libre sigue siendo un movimiento social y político; hay algunos que inclusive dicen que es ideológico.

Con el éxito de Linux, las posibilidades de colaboración de Internet, los miles de proyectos y empresas surgieron en este nuevo cosmos de software; el aspecto social pasó a un segundo plano. La apertura del propio código fuente pasó de ser un requisito técnico a una característica definitoria: tan pronto como el código fuese visible, el software se considera "código abierto". Los motivos sociales dieron paso a un enfoque más pragmático para el desarrollo de software.

El software libre y el software de código abierto trabajan en el mismo elemento, con los mismos métodos y en una comunidad mundial de individuos, proyectos y empresas. Sin embargo, como se han reunido desde diferentes direcciones — una social y otra pragmática técnica-- a veces hay conflictos que surgen cuando los resultados del trabajo conjunto no corresponden con los objetivos originales de ambos movimientos, especialmente cuando el software revela sus fuentes, pero no respeta las cuatro libertades del software libre al mismo tiempo, por ejemplo, cuando hay restricciones en la divulgación, el cambio o la conexión con otros componentes del software.

La *licencia* bajo la cual está disponible el software determina a qué condiciones está sujeto en cuanto a su uso, distribución y modificación. Debido a que los requisitos y motivos pueden ser muy diferentes, se han creado innumerables licencias en el área de FOSS. Debido al enfoque mucho más fundamental del movimiento del software libre, no es sorprendente que no se reconozca muchas licencias de código abierto como "gratuitas" y por lo tanto se rechace. Por el contrario, no es el caso, debido a que el enfoque del código abierto es mucho más pragmático.

Veamos brevemente la compleja área de las licencias.

Licencias

A diferencia de un refrigerador o un automóvil, el software no es un producto *físico*, sino *digital*, por lo tanto una empresa no puede transferir la propiedad de dicho producto vendiéndolo y cambiándolo de dueño, sino que transfiere los derechos de uso, y el usuario acepta contractualmente ese contracto. Los derechos de uso *no* se registran en la licencia de software, por tanto, es comprensible la importancia de las normas que posee.

Mientras que los grandes proveedores de software privativo, como Microsoft o SAP, tienen sus propias licencias que se adaptan con precisión a sus productos; los defensores del software libre y de código abierto se han esforzado desde el principio por la claridad y la validez general de sus licencias, porque después de todo, cada usuario debería entenderlas y si es necesario, utilizarlas para sus propios desarrollos.

Sin embargo, no se debe ocultar que este ideal de simplicidad difícilmente puede lograrse, porque hay demasiados requisitos específicos y entendimientos jurídicos que no siempre son compatibles a nivel internacional y que se oponen a ello. Por decir sólo un ejemplo, las leyes de derechos de autor alemanas y estadounidenses son fundamentalmente diferentes, de acuerdo con el derecho alemán, hay una *persona* como *autor* (más precisamente: *Urheber*), cuyo trabajo es de su *propiedad intelectual*; mientras que el autor puede conceder el permiso para utilizar su obra, no puede asignar o renunciar a su autoría. Este último es ajeno a la ley estadounidense. Aquí también hay un autor (más sin embargo, también puede ser una empresa o una institución), pero solo tiene derechos de explotación que se puede transferir en partes o en su totalidad del producto, y por lo tanto, separarse completamente de su trabajo. Una licencia internacionalmente válida debe interpretarse con respecto a una legislación diferente.

Las consecuencias son numerosas y a veces, muy diferentes por la cantidad de licencias FOSS, y lo que es peor, son las diferentes versiones de una licencia, o una mezcla de licencias (dentro de un proyecto, o incluso cuando se conectan varios proyectos) que pueden causar confusión o incluso disputas legales.

Tanto los representantes del software libre, así como los defensores del movimiento de código abierto crearon sus propias organizaciones; que hoy en día son responsables de la formulación de licencias de software de acuerdo con sus principios y apoyo a sus miembros en la realización.

Copyleft

La mencionada *Free Software Foundation* (FSF) ha formulado la *GNU General Public License* (GPL) como una de las licencias más importantes para el software libre, que ha sido utilizada por muchos proyectos, por ejemplo, el kernel de Linux. Además, ha publicado licencias con adaptaciones específicas para cada caso, como la *GNU Lesser General Public License* (LGPL) que rige la combinación de software libre con modificaciones realizadas en el código cuando estas no tienen que

ser liberadas al público, la *GNU Affero General Public License* (AGPL), que cubre la venta de acceso al software alojado, o la *GNU Free Documentation License* (FDL) que extiende los principios de la libertad a la documentación del software. Además, la FSF hace recomendaciones a favor o en contra de las licencias de terceros, y los proyectos afiliados como GPL-Violations.org investigan presuntas violaciones de las licencias libres.

La FSF nombra el principio de una licencia libre cuando se aplica las variantes del software *copyleft*—en contraste con el copyright restrictivo que lo rechaza. Por lo tanto, la idea es transferir los principios liberales de una licencia de software tan ilimitadamente como sea posible, a futuras variantes del software para evitar restricciones posteriores.

Lo que parece obvio y simple conduce a complicaciones considerables en la práctica, por está razón los críticos a menudo llaman al principio copyleft “viral”, ya que se transmite a versiones posteriores.

Tomando un ejemplo de lo que se ha dicho, se deduce que dos componentes de software que tienen licencia bajo diferentes licencias copyleft podrían no ser combinables entre sí, ya que ambas licencias no pueden transferirse a un producto posterior al mismo tiempo. ¡Esto incluso puede aplicarse a diferentes versiones de la misma licencia!

Por esta razón, las nuevas licencias o versiones de licencia a menudo ya no comprenden el copyleft con tanta rigidez. La mencionada *GNU Lesser General Public License* (LGPL) es en este sentido una concesión para poder conectar software libre con componentes “no libres” como se hace frecuentemente con las llamadas *librerías*. Las librerías contienen subrutinas o rutinas, que a su vez son utilizadas por varios otros programas.

Otra forma de evitar conflictos de licencia es *dual licensing*, donde un software es licenciado bajo diferentes licencias, por ejemplo, una licencia libre y una propietaria. Un caso típico es una versión libre de un software que sólo puede ser usado cuando se respetan las restricciones del copyleft y la oferta alternativa de obtener el software bajo una licencia diferente, lo que libera al licenciatario de cierta restricción a cambio de una cuota que podría ser utilizada para financiar el desarrollo del software.

Por lo tanto, debe quedar claro que la elección de la licencia para los proyectos de software debe hacerse con mucha cautela, ya que de ella depende la cooperación con otros proyectos, la combinación con otros componentes y también el diseño futuro del propio producto. El copyleft presenta a los desarrolladores desafíos especiales a este respecto.

Definición de Código Abierto y Licencias Permisivas

Por lo que respecta al código abierto, es la *Open Source Initiative* (OSI), fundada en 1998 por Eric S. Raymond y Bruce Perens; quién dentro su principal función verifica temas relacionados con las licencias. También desarrollaron un procedimiento estandarizado para comprobar que las licencias

de software cumplen con la *Open Source Definition*. En la actualidad se pueden encontrar más de 80 licencias de código abierto reconocidas en el sitio web de OSI.

También se enumeran las licencias como “aprobadas por la OSI” (OSI-approved) que contradicen explícitamente el principio de copyleft, especialmente el grupo *BSD licenses*. El *Berkeley Software Distribution* (BSD) es una variante del sistema operativo Unix desarrollado originalmente en la Universidad de Berkeley, que más tarde dio lugar a proyectos libres como *NetBSD*, *FreeBSD* y *OpenBSD*. Las licencias subyacentes de estos proyectos comúnmente se denominan como *permisivas*. A diferencia de las licencias copyleft, no tienen el objetivo de establecer las variantes modificadas en los términos de uso. Más bien, la máxima libertad debería ayudar a que el software se distribuya lo más ampliamente posible al dejar a los desarrolladores o editores del software decidir cómo proceder con las ediciones. Un ejemplo, es que las publiquen o las manejen como código cerrado y las distribuyan comercialmente.

La Licencia *2-Clause BSD*, también conocida como *Simplified BSD License* o *FreeBSD License*, demuestra lo reducida que puede ser esta licencia permisiva. Además, la cláusula de responsabilidad estandarizada protege a los desarrolladores de reclamos de daños causados por el software. La licencia consta sólo de las siguientes dos reglas:

Se permite la redistribución y el uso en fuentes y formas binarias, con o sin modificación, siempre y cuando que se cumplan las siguientes condiciones:

1. Las redistribuciones de código fuente deben conservar el aviso de copyright anterior, la lista de condiciones y la cláusula de exención de responsabilidad.
2. Las redistribuciones en forma binaria deben reproducir el aviso de derechos de autor anterior, la lista de condiciones y la cláusula de exención de responsabilidad en la documentación y/u otros materiales proporcionados con la distribución.

Creative Commons

El exitoso concepto del desarrollo de FLOSS y el progreso tecnológico asociado a este, cundujeron a intentos de transferir el principio de código abierto a otras áreas no técnicas. La preparación y provisión de conocimientos, así como la cooperación creativa en la resolución de tareas complejas, se consideran ahora como evidencia del principio de código abierto ampliado y relacionado con el contenido.

Todo esto condujo a la necesidad de crear bases confiables en estas áreas, que según los resultados del trabajo se pueden compartir y procesar. Dado que las licencias de software disponibles eran difícilmente adecuadas para ello, hubo numerosos intentos de convertir los requisitos específicos del trabajo científico en obras de arte digitalizadas “el espíritu del código abierto” (in the spirit of open source) en licencias igualmente prácticas.

La iniciativa más importante de este tipo hoy en día es *Creative Commons* (CC), que resume sus preocupaciones de la siguiente manera:

> Creative Commons es una organización global sin fines de lucro que permite compartir, reutilizar la creatividad y el conocimiento a través de la provisión de herramientas legales gratuitas.
>
> — https://creativecommons.org/faq/#what-is-creative-commons-and-what-do-you-do

Con Creative Commons, el enfoque de la asignación de derechos se remonta desde el distribuidor al autor, por ejemplo, en la publicación tradicional, el autor suele transferir todos los derechos de publicación (impresión, traducción, etc.) a un editor, que a su vez garantiza la mejor distribución posible de la obra; ya que los canales de distribución de Internet han cambiado significativamente, colocan el autor en posición de ejercer muchos de estos derechos de publicación y decidir por sí mismo la forma en que se puede utilizar su obra. Creative Commons brinda la oportunidad de determinar esto de manera simple y legalmente confiable, pero Creative Commons quiere más: los autores son alentados a colocar a disposición sus trabajos como contribución a un proceso general de intercambio y cooperación. A diferencia de los derechos de autor tradicionales, que otorgan al autor todos los derechos que pueden transferir a otros según sea necesario, el enfoque de Creative Commons adopta el enfoque opuesto: el autor coloca su trabajo a disposición de la comunidad, pero puede elegir entre un conjunto de características que necesita a tener en cuenta al usar su trabajo: cuantas más características elija, más restrictiva será la licencia.

Entonces, el principio de "Elegir una licencia" de CC es preguntar a un autor paso a paso las propiedades individuales y generar la licencia recomendada; así el autor puede realizar su última asignación a su trabajo como texto e ícono.

Para una mejor comprensión, aquí hay una visión general de las seis posibles combinaciones y licencias ofrecidas por CC:

CC BY ("Atribución")
: La licencia libre permite a cualquiera editar y distribuir la obra siempre que nombre al autor.

CC BY-SA ("Atribución - Compartir igual")
: Como CC BY, excepto que el trabajo modificado solo puede distribuirse bajo la misma licencia. Es el principio del copyleft, porque la licencia es "heredada" al igual que aquí.

CC BY-ND ("Atribución - No derivada")
: Como CC BY, excepto que el trabajo sólo puede ser transmitido sin modificaciones.

CC BY-NC (“Atribución - No comercial”)

La obra puede ser editada y distribuida nombrando al autor, pero sólo bajo condiciones no comerciales.

CC BY-NC-SA (“Atribución - No comercial - Compartir igual”)

Como BY-NC, excepto que el trabajo sólo puede ser compartido bajo las mismas condiciones (es decir, una licencia tipo copyleft).

CC BY-NC-ND ("Atribución - No comercial - No derivada`")

Es la licencia más restrictiva: la distribución está permitida con atribución del autor, pero sólo sin cambios y bajo condiciones no comerciales.

Modelos de Negocio en Software Libre

En retrospectiva, el triunfo del FLOSS actúa como un movimiento de base de idealistas tecnófilos que independientemente de las restricciones económicas y de las dependencias monetarias colocan su trabajo al servicio del público en general. Al mismo tiempo se han creado empresas con valores de miles de millones en el entorno del FLOSS; por citar sólo una, la empresa estadounidense *Red Hat*, fundada en 1993, con ventas anuales superiores a los 3.000 millones de dólares (2018), que a su vez fue absorbida por el gigante de las tecnologías de la información y la comunicación, IBM, en 2018.

Así que echemos un vistazo a la tensión entre la distribución gratuita del software de alta calidad y los modelos de negocio de sus creadores, porque una cosa debe quedar clara: los innumerables desarrolladores de software libre altamente cualificados también deben ganar dinero, y el entorno FLOSS originalmente no comercial debe por tanto desarrollar modelos de negocio sostenibles con el fin de preservar su propio cosmos.

Un enfoque común, especialmente en la fase inicial para proyectos grandes, es el llamado *crowdfunding*, es decir, la recolección de donaciones (dinero) a través de una plataforma como *Kickstarter*. A cambio, los donantes reciben una bonificación predefinida de los desarrolladores en caso de éxito, es decir, si alcanzan los objetivos previamente definidos. Unos ejemplos de estas bonificaciones son el acceso ilimitado al producto o a las características especiales.

Otro enfoque es *dual licensing*: el software libre se ofrece en paralelo bajo una licencia más restrictiva o incluso propietaria, que a su vez garantiza al cliente servicios más extensos (tiempos de respuesta en caso de errores, actualizaciones, versiones para determinadas plataformas, etc.). Un ejemplo entre muchos es *ownCloud*, que se está desarrollando bajo la GPL y ofrece a los clientes empresariales una “Edición Comercial” bajo una licencia propietaria.

También tomemos a ownCloud como ejemplo de otro modelo de negocio generalizado de FLOSS: los servicios profesionales. Muchas empresas carecen del conocimiento técnico interno necesario para

configurar y operar software complejo y crítico de manera confiable y, sobre todo, de forma segura. Por esa razón compran servicios profesionales como consultoría, mantenimiento o servicio de asistencia directamente del fabricante. Las cuestiones de responsabilidad también juegan un papel en esta decisión, ya que la compañía transfiere los riesgos de operación al fabricante.

Si un software logra ser exitoso y popular en su campo, sus posibilidades de monetización periférica como el merchandising o los certificados que los clientes adquieren, señalarán su estatus especial a la hora de utilizar este software. Por ejemplo, la plataforma de aprendizaje *Moodle* ofrece la certificación de entrenadores que documentan sus conocimientos a clientes potenciales, y este es sólo un ejemplo entre muchos otros.

El software como servicio *Software as a Service* (SaaS) es otro modelo de negocio, especialmente para las tecnologías basadas en web, donde un proveedor de cloud computing ejecuta un software como el CRM (Customer Relationship Management) o el CMS (Content Management System) en sus servidores y concede a sus clientes acceso a la aplicación instalada. Esto le ahorra al cliente la instalación y el mantenimiento del software, que a su vez paga por el uso del aplicativo en función de diversos parámetros, por ejemplo, el número de usuarios, donde la disponibilidad y la seguridad juegan un papel importante como factores críticos para el negocio.

Por último, pero no menos importante, el modelo de desarrollo de extensiones por solicitud del cliente en software libre. Por lo general, esto son proyectos más pequeños y depende del cliente decidir cómo proceder con estas extensiones, es decir, si también las libera o las mantiene bajo llave como parte de su propio modelo de negocio.

Una cosa debería haber quedado clara: Aunque el software libre suele estar disponible de forma gratuita, se han creado numerosos modelos de negocio en su entorno que son constantemente modificados y ampliados por innumerables autónomos y empresas de todo el mundo de forma muy creativa; lo que en última instancia garantiza la existencia continuada de todo el movimiento FLOSS.

Ejercicios guiados

1. ¿Cuáles son, en pocas palabras, las "cuatro libertades" definidas por Richard Stallman y la Free Software Foundation?

libertad 0	
libertad 1	
libertad 2	
libertad 3	

2. ¿Qué significa la abreviatura FLOSS?

3. Ha desarrollado software libre y desea asegurar el software en sí, pero que también todos los trabajos futuros basados en este permanezcan libres. ¿Qué licencia eliges?

CC BY	
GPL version 3	
2-Clause BSD License	
LGPL	

4. ¿A cuál de las siguientes licencias llamaría permisiva, cuál llamaría copyleft?

Simplified BSD License	
GPL version 3	
CC BY	
CC BY-SA	

5. Ha escrito una aplicación web y la ha publicado con una licencia gratuita. ¿Cómo puedes ganar dinero con tu producto? Nombra tres posibilidades.

Ejercicios exploratorios

1. ¿Bajo qué licencia (incluida la versión) están disponibles las siguientes aplicaciones?

Apache HTTP Server	
MySQL Community Server	
Wikipedia articles	
Mozilla Firefox	
GIMP	

2. Desea lanzar su software bajo la GNU GPL v3. ¿Qué pasos debes seguir?

3. Usted ha un escrito software propietario y desea combinarlo con software libre bajo GPL versión 3. ¿Se le permite hacer esto o qué debe considerar?

4. ¿Por qué la Free Software Foundation lanzó la *GNU Affero General Public License* (GNU AGPL) como complemento a la GNU GPL?

5. Nombre tres ejemplos de software libre, que también se ofrecen como “Business Edition”, es decir, en una versión de pago.

Resumen

En esta lección has aprendido:

- Similitudes y diferencias entre software libre y de código abierto (FLOSS)
- Licencias de software libre, su importancia y problemas
- Copyleft vs. licencias permisivas
- Modelos de negocio FLOSS

Respuestas a los ejercicios guiados

1. ¿Cuáles son, en pocas palabras, las "cuatro libertades" definidas por Richard Stallman y la Free Software Foundation?

libertad 0	ejecuta el software
libertad 1	estudiar y modificar el software (código fuente)
libertad 2	distribuir el software
libertad 3	distribuir el software modificado

2. ¿Qué significa la abreviatura FLOSS?

   Free/Libre Open Source Software

3. Ha desarrollado software libre y desea asegurarse el software en sí, pero que también todos los trabajos futuros basados en este permanezcan libres. ¿Qué licencia eliges?

CC BY	
GPL version 3	X
2-Clause BSD License	
LGPL	

4. ¿A cuál de las siguientes licencias llamaría permisiva, cuál llamaría copyleft?

Simplified BSD License	permisiva
GPL version 3	copyleft
CC BY	permisiva
CC BY-SA	copyleft

5. Ha escrito una aplicación web y la ha publicado con una licencia gratuita. ¿Cómo puedes ganar dinero con tu producto? Nombra tres posibilidades.

   - Licencia doble, ejemplo, ofrecer una "Edición comercial" de pago.
   - Ofreciendo alojamiento, servicio y soporte.
   - Desarrollo de extensiones patentadas para clientes.

Respuestas a los ejercicios exploratorios

1. ¿Bajo qué licencia (incluida la versión) están disponibles las siguientes aplicaciones?

Apache HTTP Server	Apache License 2.0
MySQL Community Server	GPL 2.0
Wikipedia articles (English)	Creative Commons Attribution Share-Alike license (CC-BY-SA)
Mozilla Firefox	Mozilla Public License 2.0
GIMP	LGPL 3

2. Desea lanzar su software bajo la GNU GPL v3. ¿Qué pasos debes seguir?

   - Si es necesario, protéjase contra el empleador con una exención de derechos de autor, por ejemplo, que usted pueda especificar la licencia.
   - Agregue un aviso de copyright a cada archivo.
   - Agregue un archivo llamado “COPYING” con el texto completo de la licencia a su software.
   - Agregue una referencia a la licencia en cada archivo.

3. Usted ha escrito un software propietario y desea combinarlo con software libre bajo GPL versión 3. ¿Se le permite hacer esto o qué debe considerar?

   Las FAQs de la Free Software Foundation proporcionan información aquí: Siempre que su software propietario y el software libre permanezcan separados entre sí, la combinación es posible. Sin embargo, debe asegurarse de que esta separación esté técnicamente garantizada y sea reconocible para los usuarios. Si integra el software libre de tal manera que se convierta en parte de su producto, también debe publicar el producto bajo la GPL de acuerdo con el principio copyleft.

4. ¿Por qué la Free Software Foundation lanzó la *GNU Affero General Public License* (GNU AGPL) como suplemento a la GNU GPL?

   GNU AGPL cierra una brecha de licencia que surge especialmente con el software gratuito alojado en un servidor: si un desarrollador realiza cambios en el software, no está obligado por la GPL a hacer accesibles estos cambios, ya que permite el acceso, pero no “redistribuye” el programa en licencia GPL. GNU AGPL, por otro lado, estipula que el software debe estar disponible para descargar con todos los cambios.

5. Nombre tres ejemplos de software libre, que también se ofrecen como “Business Edition”, es

decir, en una versión de pago.

MySQL, Zammad, Nextcloud



1.4 Destrezas TIC y el trabajo con Linux

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 1.4

Importancia

2

Áreas de conocimiento clave

- Destrezas en el escritorio
- Iniciación a la línea de comandos
- Usos de Linux en la Industria, Cloud Computing y Virtualización

Lista parcial de archivos, términos y utilidades

- Uso del navegador, preocupaciones en torno a la privacidad, opciones de configuración, búsquedas en la web y almacenamiento de contenido
- La terminal y la consola
- Problemas relacionados con las contraseñas
- Problemas y herramientas relacionadas con la privacidad
- Uso de aplicaciones de código abierto comunes en presentaciones y proyectos



1.4 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	1 La comunidad Linux y una carrera en el Open Source
Objetivo:	1.4 Habilidades TIC y trabajando en Linux
Lección:	1 de 1

Introducción

Hubo un tiempo en el que trabajar con Linux en ordenadores de escritorio se consideraba difícil, ya que el sistema carecía de muchas aplicaciones importantes y de herramientas de configuración que tenían otros sistemas operativos. Algunas de las razones fueron, que Linux era más nuevo que muchos otros sistemas operativos; por lo que era más fácil empezar a desarrollar las aplicaciones más esenciales en línea de comandos y dejar las herramientas gráficas más complejas para más adelante. Al principio, Linux estaba dirigido a los usuarios más avanzados, por lo cual esto no debió considerarse un problema. Hoy en día, los entornos de escritorio de Linux son muy maduros y no dejan nada que desear en cuanto a características y facilidad de uso. Sin embargo, la línea de comandos todavía se considera una herramienta poderosa utilizada todos los días por usuarios avanzados. En esta lección, veremos algunas de las habilidades básicas de escritorio que necesitará para elegir la mejor herramienta para el trabajo, incluida la línea de comandos.

Interfaces de usuario en Linux

Cuando utilizas un sistema Linux, puedes interactuar con una terminal (línea de comando) o con una interfaz gráfica de usuario, ambas formas le otorgan acceso a numerosas aplicaciones que permiten

realizar casi cualquier tarea con la computadora. Si bien en el objetivo 1.2 ya se presentó una serie de aplicaciones de uso común, comenzaremos esta lección con un análisis más detallado de los entornos de escritorio, las formas de acceder al terminal y a las herramientas utilizadas para las presentaciones y la gestión de proyectos.

Entornos de escritorio

Linux tiene un enfoque modular en el que diferentes partes del sistema son desarrolladas por diferentes proyectos y desarrolladores, cada una satisface una necesidad u objetivo específico. Es por esta razón, que existen varias opciones de entornos de escritorio para elegir, junto con los gestores de paquetes. El entorno de escritorio es una de las principales diferencias entre las muchas distribuciones que existen; a diferencia de los sistemas operativos propietarios como Windows y macOS, los usuarios están restringidos al entorno de escritorio por defecto del sistema operativo, pero en Linux existe la posibilidad de instalar múltiples entornos y elegir el que más se adapte a sus necesidades.

Básicamente, hay dos entornos de escritorio principales en el mundo Linux: *Gnome* y *KDE*. Ambos son muy completos, con una gran comunidad detrás de ellos y apuntan al mismo objetivo, pero con enfoques ligeramente divergentes. En pocas palabras, Gnome intenta seguir el principio KISS (“keep it simple stupid”), con aplicaciones muy racionalizadas y limpias. Por otro lado, KDE tiene otra perspectiva con una selección más amplia de aplicaciones y da al usuario la oportunidad de cambiar cada configuración del entorno.

Mientras que las aplicaciones Gnome están basadas en el kit de herramientas (toolkit) GTK (escrito en lenguaje C); las aplicaciones KDE utilizan la librería Qt (escrita en C++). Uno de los aspectos más prácticos de escribir aplicaciones con el mismo toolkit gráfico es que las aplicaciones tenderán a compartir un aspecto similar, brindando así al usuario una sensación de unidad durante su experiencia. Otra característica importante, es disponer de la misma librería gráfica compartida para muchas de las aplicaciones de uso más frecuente, ya que esto permite ahorrar algo de espacio en la memoria y a su vez agilizar el tiempo de carga cuando la librería se ha cargado por primera vez.

Llegando a la línea de comando

Para nosotros, una de las aplicaciones más importantes es el emulador de terminal gráfico. Son llamados emuladores de terminal porque realmente emulan en un entorno gráfico. Los terminales seriales de estilo antiguo (a menudo máquinas de teletipo) en realidad eran clientes que solían estar conectados a una máquina remota donde realmente ocurría la informática. Esas máquinas eran computadoras realmente simples sin gráficos que se utilizaron en las primeras versiones de Unix.

En Gnome, tal aplicación se llama *Gnome Terminal*, mientras que en KDE se conoce como *Konsole*. Existen varias opciones disponibles, como *Xterm*. Esas aplicaciones permite que tengamos acceso a

un entorno de línea de comandos para poder interactuar con la shell.

Por lo tanto, debe mirar el menú de aplicaciones de su distribución para encontrar la aplicación de terminal. Además de cualquier diferencia entre ellas, cada aplicación le ofrecerá lo necesario para ganar confianza en el uso de la línea de comandos.

Otra forma de entrar en el terminal es utilizar la TTY virtual. Usted puede entrar pulsando `Ctrl` + `Alt` + `F#`. Comprender que `F#` es una de las teclas de función del 1 al 7. Probablemente, algunas de las combinaciones iniciales pueden ser la ejecución de su gestor de sesiones o su entorno gráfico, mientras que otras mostrarán un mensaje solicitando su nombre de usuario, como por ejemplo, el que aparece a continuación:

```
Ubuntu 18.10 arrelia tty3
arrelia login:
```

En este caso `arrelia` es el nombre de host de la máquina y la `tty3` es la terminal disponible después de usar la combinación de teclas anterior más la tecla `F3`, como en `Ctrl` + `Alt` + `F3`.

Después de proporcionar su nombre de usuario y contraseña, podrá entrar a una terminal, pero recuerde que no hay un entorno gráfico, por lo que no podrá usar el ratón o ejecutar aplicaciones gráficas sin antes iniciar una sesión X, o Wayland. Sin embargo, eso está fuera del alcance de esta lección.

Presentaciones y proyectos

La herramienta más importante para las presentaciones en Linux es *LibreOffice Impress*. Es parte de la suite ofimática de código abierto llamada *LibreOffice*. Piense en LibreOffice como un reemplazo de código abierto para el equivalente a *Microsoft Office*. Puede incluso abrir y guardar los archivos PPT y PPTX que son nativos de *Powerpoint*, pero a pesar de eso, realmente recomiendo que use el formato nativo ODP Impress. El ODP es parte del *Open Document Format* ; este es un estándar internacional para este tipo de archivo. Es especialmente importante si desea mantener sus documentos accesibles durante muchos años y preocuparse menos por los problemas de compatibilidad, ya que, al ser un estándar abierto, es posible que cualquiera pueda implementar el formato sin pagar regalías ni licencias. Esto también le permite probar otros softwares de presentaciones que le gusten más y llevar sus archivos consigo, ya que es muy probable que sean compatibles con esos aplicativos más nuevos.

Pero si prefiere el código sobre las interfaces gráficas, hay algunas herramientas a elegir. *Beamer* es una clase *LaTeX* que puede crear presentaciones de diapositivas a partir de su código. LaTeX es un sistema de composición tipográfica muy utilizado para escribir documentos científicos en la academia, especialmente por su capacidad para manejar símbolos matemáticos complejos, con los

que otros softwares tienen dificultades para lidiar. Si estás en la universidad y necesitas lidiar con ecuaciones y otros problemas relacionados con las matemáticas, Beamer puede ahorrarte mucho tiempo.

La otra opción es *Reveal.js*, un impresionante paquete NPM (NPM es el gestor de paquetes NodeJS por defecto) que te permite crear hermosas presentaciones usando la web. Así que, si puedes escribir HTML y CSS, Reveal.js traerá la mayor parte del JavaScript necesario para crear bonitas e interactivas presentaciones que se adaptarán bien a cualquier resolución y tamaño de pantalla.

Por último, si desea sustituir *Microsoft Project*, puede probar *GanttProject* o *ProjectLibre*. Ambos son muy similares a sus equivalentes propietarios y compatibles con los archivos de proyecto.

Usos industriales de Linux

Linux es muy utilizado entre las industrias de software e Internet. Sitios como W3Techs reportan que cerca del 68% de los servidores de sitios web en Internet son impulsados por Unix y la mayor parte de ellos son conocidos por ser Linux.

Esta gran adopción se da no sólo por la naturaleza libre de Linux (tanto en la cerveza libre como en la libertad de expresión) sino también por su estabilidad, flexibilidad y rendimiento. Estas características permiten a los proveedores ofrecer sus servicios a un coste menor y mayor escalabilidad. Una parte importante de los sistemas Linux se ejecutan hoy en día en la nube, ya sea en un modelo IaaS (Infraestructura como servicio), PaaS (Plataforma como servicio) o SaaS (Software como servicio).

IaaaS es una forma de compartir los recursos de un gran servidor ofreciéndoles acceso a máquinas virtuales, de hecho, son múltiples sistemas operativos que se ejecutan como invitados en una máquina anfitriona a través de un software importante que se llama *hipervisor*. El hipervisor es responsable de hacer posible que estos SO invitados se ejecuten separando y administrando los recursos disponibles en la máquina anfitriona para esos invitados. Esto es lo que llamamos virtualización. En el modelo IaaS, paga solo por la fracción de recursos que usa su infraestructura.

Linux tiene tres conocidos hipervisores de código abierto: *Xen*, *KVM* y *VirtualBox*. Xen es probablemente el más antiguo de ellos, KVM ha dejado de lado a Xen como el más prominente de los hipervisores de Linux, tiene su desarrollo patrocinado por RedHat y es utilizado por ellos y por otros usuarios, tanto en servicios de nube públicas como en configuraciones de nube privadas. VirtualBox pertenece a Oracle desde su adquisición de Sun Microsystems y suele ser utilizado por los usuarios finales debido a su facilidad de uso y administración.

Por otro lado, PaaS y SaaS se basan en el modelo IaaS, tanto desde el punto de vista técnico como conceptual. En PaaS, en lugar de una máquina virtual, los usuarios tienen acceso a una plataforma en la que es posible desplegar y ejecutar su aplicación. El objetivo de aliviar la carga de las tareas de

administración del sistema y las actualizaciones de los sistemas operativos. Heroku es un ejemplo común de PaaS en el que el código de programa puede ejecutarse sin tener que ocuparse de los contenedores subyacentes y las máquinas virtuales.

Por último, el SaaS es el modelo en el que normalmente se paga una suscripción para utilizar un software sin preocuparse de nada más. *Dropbox* y *Salesforce* son dos buenos ejemplos de SaaS. La mayoría de estos servicios se acceden a través de un navegador web.

Un proyecto como *OpenStack* es una colección de software de código abierto que puede hacer uso de diferentes hipervisores y otras herramientas con el fin de ofrecer un entorno completo de IaaS en la nube aprovechando la potencia del clúster informático en su propio centro de datos. Sin embargo, la configuración de dicha infraestructura no es trivial.

Problemas de privacidad en el uso de Internet

En estos días el navegador web es una pieza fundamental de software en cualquier escritorio, pero algunas personas aún carecen del conocimiento para usarlo de forma segura. Si bien se accede a más y más servicios a través de un navegador web, casi todas las acciones realizadas a través de un navegador son rastreadas y analizadas por varias partes. Asegurar el acceso a los servicios de Internet y prevenir el seguimiento es un aspecto importante del uso de Internet de manera segura.

Cookie Tracking

Supongamos que ha navegado por un sitio web de comercio electrónico, ha seleccionado un producto que deseaba y lo ha colocado en el carrito de compra, pero en el último momento, ha decidido pensárselo dos veces y reflexionar si realmente lo necesita. Después de un tiempo, comienza a ver anuncios del mismo producto siguiéndolo por la web, al hacer clic en ellos, lo redirecciona nuevamente a la página de la tienda. No es raro que los productos que colocaste en el carrito de compras todavía estén allí esperando que decidas echarles un vistazo. ¿Alguna vez te has preguntado cómo lo hacen? ¿Cómo le muestran el anuncio correcto en otra página web? La respuesta a esta pregunta se llama *cookie tracking*.

Las cookies son pequeños archivos que un sitio web guarda en su ordenador para almacenar y recuperar algún tipo de información, que sea útil para su navegación, Se utilizan desde hace muchos años y son una de las formas más antiguas de almacenar datos del lado del cliente; un buen ejemplo de su uso son los identificadores únicos de las tarjetas de compra, de esta forma, si alguna vez vuelve al mismo sitio web en unos días, la tienda puede recordarle los productos que ha colocado en su carrito durante su última visita y ahorrarle tiempo para encontrarlos de nuevo.

Eso suele estar bien, ya que el sitio web te ofrece una función útil y no comparte datos con terceros. Pero ¿qué pasa con los anuncios que te muestran mientras navegas en otras páginas web? Aquí es

donde entran en juego las redes publicitarias. Por un lado, las redes publicitarias son compañías que ofrecen anuncios de sitios de comercio electrónico como el de nuestro ejemplo; y por otro lado ofrecen la monetización de sitios web. Los creadores de contenido, como los blogueros, pueden hacer espacio disponible para esas redes publicitarias en su blog, a cambio de una comisión relacionada con las ventas generadas por ese anuncio.

Pero ¿cómo saben qué producto mostrarles? Por lo general, lo hacen guardando una cookie de la red publicitaria en el momento en que visitó o buscó un determinado producto en el sitio web del comercio electrónico. Al hacer esto, la red puede recuperar la información de esa cookie donde quiera que se encuentre los anuncios, haciendo la correlación con los productos que le interesaban. Esta suele ser una de las formas más comunes de rastrear a alguien por Internet. El ejemplo que dimos anteriormente hace uso de un comercio electrónico para hacer las cosas más tangibles, pero las plataformas de redes sociales hacen lo mismo con sus botones “Like” o “Share” y su inicio de sesión social.

Una forma de deshacerse de eso es al no permitir que sitios web de terceros almacenen cookies en su navegador. De esta manera, solo el sitio web que visita puede almacenar sus cookies, pero tenga en cuenta que algunas características “legítimas” pueden no funcionar bien si lo hace, porque muchos sitios hoy en día dependen de servicios de terceros para funcionar. Por lo tanto, puede buscar un administrador de cookies en el repositorio de complementos de su navegador para tener un control detallado de qué cookies se almacenan en su máquina.

Do Not Track (DNT)

Otro error común está relacionado con una configuración del navegador mejor conocida como DNT, que es un acrónimo de “No rastrear” siglas en Ingles (Do Not Track) y que puede ser activado básicamente en cualquier navegador actual. Al igual que en el modo privado, no es difícil encontrar personas que creen que no serán rastreadas si tienen esta configuración activada, desafortunadamente, eso no siempre es cierto. Actualmente, DNT es solo una forma de decirle a los sitios web que visita que no desea que lo rastreen. Pero, de hecho, ellos son los que decidirán si respetarán su elección o no. En otras palabras, DNT es una forma de darse de baja del seguimiento del sitio web, pero no hay garantía sobre esa elección.

Técnicamente, esto se hace simplemente enviando un indicador adicional en el encabezado del protocolo de petición HTTP ('DNT: 1') al solicitar datos de un servidor web. Si desea saber más sobre este tema, el sitio web https://allaboutdnt.com es un buen punto de partida.

“Private” Windows

Es posible que haya notado las citas en el encabezado anterior. Esto se debe a que esas ventanas no son tan privadas como la mayoría de la gente piensa, los nombres pueden variar, pero pueden

llamarse “modo privado”, “incógnito” o “anónimo” dependiendo del navegador que estés usando.

En Firefox, puedes usarlo fácilmente pulsando las teclas `Ctrl` + `Shift` + `P`. En Chrome, sólo tienes que pulsar `Ctrl` + `Shift` + `N`. Lo que realmente hacen es abrir una nueva sesión, que normalmente no comparte ninguna configuración o datos de tu perfil estándar. Cuando cierras la ventana privada, el navegador borrará automáticamente todos los datos generados por esa sesión, sin dejar rastro en el ordenador utilizado, lo que significa que no se almacenan datos personales, como el historial, las contraseñas o las cookies en ese ordenador.

Por lo tanto, muchas personas malinterpretan este concepto al creer que pueden navegar de forma anónima en Internet, lo cual no es completamente cierto. Una de las cosas que hace el modo de privado o incógnito es evitar el llamado rastreo de cookies. Cuando usted visita un sitio web, puede almacenar un pequeño archivo en su computadora que puede contener una identificación que será usada para rastrearlo. A menos que usted configure su navegador para que no acepte cookies de terceros; las redes de publicidad u otras compañías podrán almacenar y recuperar esa identificación y rastrear su navegación a través de los sitios web.

Además de eso, los sitios web y otros pares en Internet aún pueden usar muchas otras técnicas para rastrearlo. Entonces, el modo privado le brinda cierto nivel de anonimato, pero es completamente privado en la computadora que esté utilizando. Si accede a su cuenta de correo electrónico o sitio web bancario desde una computadora pública, como en un aeropuerto o un hotel, definitivamente debe accederlos utilizando el modo privado de su navegador. En otras situaciones, puede haber beneficios, pero debe saber exactamente qué riesgos está evitando y cuáles no tienen ningún efecto. Siempre que use una computadora accesible al público, tenga en cuenta que pueden existir otras amenazas de seguridad como malwares o registradores de teclas (Key Loggers). Tenga cuidado cada vez que ingrese información personal, incluidos nombres de usuario y contraseñas, en dichas computadoras o cuando descargue o copie datos confidenciales.

Eligiendo la Contraseña Correcta

Una de las situaciones más difíciles a las que se enfrenta cualquier usuario es la de elegir una contraseña segura para los servicios de los que hace uso, ya que seguramente ha escuchado que no debe usar combinaciones comunes como `qwerty`, `123456` o `654321`, ni números fácilmente adivinables como su cumpleaños (o el de un pariente) o el código postal. La razón de esto es porque son combinaciones muy evidentes y los primeros intentos de un invasor por acceder a su cuenta son muy obvias.

Hay técnicas conocidas para crear una contraseña segura. Una de las más famosas es inventar una frase que te recuerde ese servicio, y elegir las primeras letras de cada palabra. Supongamos que quiere crear una buena contraseña para Facebook, por ejemplo, podría crear una frase como “Sería feliz si tuviera 1000 amigos como Mike”, escoge la primera letra de cada palabra y la contraseña final

sería `IwbhiIha1000flM`. Esto daría como resultado una contraseña de 15 caracteres, lo cual es suficientemente difícil de adivinar y fácil de usar.

Las frases suelen ser más fáciles de recordar que las contraseñas, pero incluso este método tiene sus limitaciones. Hoy en día tenemos que crear contraseñas para tantos servicios y como las usamos con diferentes frecuencias, eventualmente será muy difícil recordar todas las frases en el momento en que las necesitemos. Entonces, ¿qué podemos hacer? puede responder que lo más inteligente en este caso es crear un par de contraseñas buenas y reutilizarlas en servicios similares, ¿verdad?

Lamentablemente, tampoco es una buena idea. Probablemente también haya escuchado que no debe reutilizar la misma contraseña entre diferentes servicios. El problema de hacer tal cosa es que un servicio específico puede filtrar tu contraseña (sí, sucede todo el tiempo) y cualquier persona que tenga acceso a él intentará usar la misma combinación de correo electrónico y contraseña en otros servicios populares de Internet con la esperanza de que hayas hecho exactamente eso: Reciclar las contraseñas. ¿Y adivina qué? En caso de que tengan razón, terminará teniendo un problema no solo en un solo servicio sino en varios de ellos. Y créeme, tendemos a pensar que no nos va a pasar hasta que sea demasiado tarde.

Entonces, ¿qué podemos hacer para protegernos?, uno de los enfoques más seguros disponibles hoy en día es el uso de un *Gestor de contraseñas*. Los gestores de contraseñas son un software que esencialmente almacena todas sus contraseñas y nombres de usuario en un formato encriptado que puede ser descifrado por una contraseña maestra. De esta manera, solo necesita recordar una buena contraseña, ya que el administrador mantendrá todas las demás seguras para usted.

KeePass es uno de los administradores de contraseñas de código abierto más famosos y con más funciones disponibles. Almacenará sus contraseñas en un archivo cifrado dentro de su sistema de archivos. El hecho de que sea de código abierto es un tema importante para este tipo de software, ya que garantiza que no utilizarán sus datos debido a que cualquier desarrollador puede auditar el código y saber exactamente cómo funciona. De esta manera brinda un nivel de transparencia que es imposible de alcanzar con código propietario. KeePass tiene puertos para la mayoría de los sistemas operativos, incluidos Windows, Linux y macOS; así como dispositivos móviles como iOS y Android. También incluye un sistema de complemento que puede extender su funcionalidad mucho más allá de los valores predeterminados.

Bitwarden es otra solución de código abierto que tiene un enfoque similar, pero en lugar de almacenar sus datos en un archivo, utilizará un servidor en la nube. De esta manera es más fácil mantener todos sus dispositivos sincronizados y sus contraseñas fácilmente accesibles a través de la web. *Bitwarden* es uno de los pocos proyectos que hará que no sólo los clientes, sino también el servidor en la nube esté disponible como un software de código abierto, lo que significa que usted puede alojar su propia versión de Bitwarden y ponerla a disposición de cualquier persona, por ejemplo, su familia o los empleados de su empresa. Esto le dará flexibilidad, pero también un control

total sobre cómo se almacenan y usan sus contraseñas.

Una de las cosas más importantes para tener en cuenta cuando se utiliza un administrador de contraseñas, es crear una clave aleatoria para cada servicio diferente, ya que de todos modos no tendrá que recordarlas. Sería inútil si usa un administrador de contraseñas para almacenar contraseñas recicladas o fácilmente adivinables. Por lo tanto, la mayoría de ellos le ofrecerán un generador de contraseñas aleatorias que puede usar para crearlos.

Encriptación

Cada vez que se transfieren o almacenan datos, se deben tomar precauciones para garantizar que terceros no puedan acceder a ellos. Los datos transferidos a través de Internet pasan por una serie de enrutadores y redes en los que terceros pueden acceder al tráfico de la red. Del mismo modo, los datos almacenados en medios físicos pueden ser leídos por cualquier persona que tome posesión de esos medios. Para evitar este tipo de acceso, la información confidencial debe encriptarse antes de que salga de un dispositivo informático.

TLS

Transport Layer Security (TLS) es un protocolo que ofrece seguridad a través de conexiones de red mediante el uso de criptografía. TLS es el sucesor de *Secure Sockets Layer* (SSL) que ha quedado en desuso debido a sus fallas graves en seguridad. Además, TLS ha evolucionado un par de veces para adaptarse y ser más seguro, por lo que su versión actual es la 1.3. Está versión puede proporcionar tanto privacidad como autenticidad al hacer uso de la denominada criptografía simétrica y de su clave pública. Al decir esto, queremos decir que usted puede estar seguro de que nadie podrá espiar o alterar su comunicación con ese servidor durante esa sesión.

La lección más importante aquí es reconocer que un sitio web es confiable, por lo que debe buscar el símbolo “lock” en la barra de direcciones del navegador. Si lo desea, puede hacer clic en el navegador para inspeccionar el certificado que juega un papel importante en el protocolo HTTPS.

TLS es lo que se utiliza en el protocolo HTTPS (*HTTP over TLS*) para hacer posible el envío de datos confidenciales (como el número de su tarjeta de crédito) a través de la web. La explicación del funcionamiento de TLS va mucho más allá del propósito de este artículo, pero puede encontrar más información en Wikipedia y en Mozilla wiki.

Cifrado de archivos y correo electrónico con GnuPG

Hay muchas herramientas para asegurar los correos electrónicos, pero ciertamente una de las más importantes es *GnuPG*. GnuPG significa *GNU Privacy Guard* y es una implementación de código abierto de *OpenPGP* que es un estándar internacional codificado dentro del RFC 4880.

GnuPG puede ser utilizado para firmar, cifrar y descifrar textos, correos electrónicos, archivos, directorios e incluso particiones de disco completas. Funciona con criptografía de clave pública y está ampliamente disponible; en pocas palabras, GnuPG crea un par de archivos que contienen sus claves públicas y privadas; como su nombre lo indica, la clave pública puede estar disponible para cualquier persona y la clave privada debe mantenerse en secreto. Las personas usarán su clave pública para cifrar datos que solo su clave privada podrá descifrar.

También puede usar su clave privada para firmar cualquier archivo o correo electrónico que pueda validarse con la clave pública correspondiente. Esta señalización digital funciona de manera análoga a la firma del mundo real. Mientras sea el único que posea su clave privada, el receptor puede estar seguro de que fue usted quien la creó. Al hacer uso de la funcionalidad hash criptográfica, GnuPG también garantizará que no se hayan realizado cambios después de la firma, ya que cualquier cambio en el contenido la invalidaría.

GnuPG es una herramienta muy poderosa y en cierta medida, también compleja. Podrás encontrar más información en sitio web y en Archlinux wiki (Archlinux wiki es una muy buena fuente de información, aunque no utilices Archlinux).

Encriptación del disco

Una buena manera de proteger sus datos es cifrar todo el disco o partición. Hay muchos programas de código abierto que puede utilizar para lograr tal propósito.La forma en que funcionan y qué nivel de cifrado ofrecen también varía significativamente.Hay dos métodos básicos disponibles: *stacked* and *block* device encryption.

El cifrado de sistemas de arhivos apilados (Stacked), se implementan sobre el sistema de archivos existente. Cuando se utiliza, los archivos y directorios se cifran antes de ser almacenados en el sistema de archivos y se descifran después de leerlos. Esto significa que los archivos se almacenan en el sistema de archivos del host de forma cifrada (lo que significa que su contenido, y normalmente sus nombres de archivos/carpetas, se sustituyen por datos de apariencia aleatoria), pero aparte de eso, siguen existiendo en el sistema de archivos como si no estuvieran cifrados, como los archivos normales, los enlaces simbólicos(symlinks), enlaces duros (hardlinks), etc.

Por otro lado, el cifrado del dispositivo de bloque(block) ocurre debajo de la capa del sistema de archivos, asegurándose de que todo lo que se escribe en un dispositivo de bloque esté cifrado. Si observa el bloque mientras está fuera de línea, podrá ver una gran sección de datos aleatorios donde ni siquiera podrá detectar el tipo de sistema de archivos que hay sin antes descifrarlo. Esto significa, que no puede saber cuál es un archivo o directorio; qué tan grandes son o qué tipo de datos son, porque los metadatos, la estructura de directorios y los permisos también estarán encriptados.

Ambos métodos tienen sus propios pros y contras. Entre todas las opciones disponibles, debería echar un vistazo a *dm-crypt*, que es el estándar de facto para la encriptación de bloques para sistemas

Linux, ya que es nativo del kernel y se puede usar con la extensión *LUKS* (*Linux Unified Key Setup*), que es una especificación que implementa un estándar independiente de la plataforma para su uso con varias herramientas.

Si desea probar un método apilable, debe echar un vistazo a *EncFS*, que es probablemente la forma más fácil de proteger datos en Linux porque no requiere privilegios de root para implementar y puede funcionar en un sistema de archivos existente sin modificaciones.

Finalmente, si necesitas acceder a datos en varias plataformas, visita Veracrypt, que es el sucesor de un Truecrypt y permite la creación de medios y archivos encriptados que pueden ser usados tanto en Linux como en macOS y Windows.

Ejercicios guiados

1. Debe usar una “ventana privada” en su navegador si desea:

Para navegar completamente anónimo en Internet	
Para no dejar rastro en la computadora que está utilizando	
Para activar TLS y evitar el seguimiento de cookies	
Para usar DNT	
Para utilizar la criptografía durante la transmisión de datos	

2. ¿Qué es OpenStack?

Un proyecto que permite la creación de IaaS privadas	
Un proyecto que permite la creación de PaaS privadas	
Un proyecto que permite la creación de SaaS privadas	
Un hipervisor	
Un administrador de contraseñas de código abierto	

3. ¿Cuál de las siguientes opciones son softwares válidos de cifrado de disco ?

RevealJS, EncFS y dm-crypt	
dm-crypt y KeePass	
EncFS y Bitwarden	
EncFS y dm-crypt	
TLS y dm-crypt	

4. Seleccione verdadero o falso para el cifrado de dispositivo dm-crypt:

Los archivos se cifran antes de escribirse en el disco	
Todo el sistema de archivos es un blob cifrado	
Solo los archivos y directorios están encriptados, no los enlaces simbólicos	
No requiere acceso root	
Es un cifrado de dispositivo de bloque	

5. Beamer es:

Un mecanismo de encriptación	
Un hipervisor	
Un software de virtualización	
Un componente OpenStack	
Una herramienta de presentación de LaTeX	

Ejercicios exploratorios

1. La mayoría de las distribuciones vienen con Firefox instalado por defecto (si el tuyo no, tendrás que instalarlo primero).Vamos a instalar una extensión de Firefox llamada *Lightbeam*.Puede hacerlo pulsando `Ctrl` + `Shift` + `A` y buscando "'Lightbeam'" en el campo de búsqueda que se mostrará en la pestaña abierta, o visitando la página de extensión con Firefox y haciendo clic en el botón "'Instalar'": https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/.Después de hacer esto, inicie la extensión haciendo clic en su icono y comience a visitar algunas páginas web en otras pestañas para ver qué sucede.

2. ¿Qué es lo más importante cuando se usa un administrador de contraseñas?

3. Use su navegador web para navegar a https://haveibeenpwned.com/. Descubra el propósito del sitio web y verifique si su dirección de correo electrónico se incluyó en algunas filtraciones de datos.

Resumen

El terminal es una forma poderosa de interactuar con el sistema, y hay muchas herramientas útiles y muy maduras para usar en este tipo de entorno. Puede llegar al terminal buscando uno gráfico en el menú del entorno de su escritorio o presionando `Ctrl` + `Alt` + `F#`.

Linux se usa en gran medida en la industria tecnológica para ofrecer servicios IaaS, PaaS y SaaS. Hay tres hipervisores principales que juegan un papel importante en el soporte de esos: Xen, KVM y Virtualbox.

El navegador es una pieza esencial de software en la informática hoy en día, pero es necesario entender algunas cosas para usarlo con seguridad. DNT es sólo una manera de decirle al sitio web que usted no quiere ser rastreado, pero no hay ninguna garantía en eso.Las ventanas privadas son privadas solo para el equipo que está utilizando, pero esto puede permitirle escapar del seguimiento de cookies exactamente por eso.

TLS puede cifrar su comunicación en Internet, pero debe poder reconocer cuándo está en uso. El uso de contraseñas seguras también es muy importante para mantenerlo a salvo, por lo que la mejor idea es delegar esa responsabilidad en un administrador de contraseñas y permitir que el software cree contraseñas aleatorias en cada sitio en el que inicie sesión.

Otra forma de asegurar su comunicación es firmar y cifrar sus carpetas de archivos y correos electrónicos con GnuPG. dm-cypt y EncFS son dos alternativas para cifrar discos enteros o particiones que usan métodos de cifrado de bloques y pilas respectivamente.

Finalmente, LibreOffice Impress es una alternativa de código abierto muy completa a Microsoft Powerpoint, pero existen otras opciones como Beamer y RevealJS si prefieres crear presentaciones usando código en lugar de GUI. ProjectLibre y GanttProject pueden ser la opción correcta si necesita un reemplazo de Microsoft Project.

Respuestas a los ejercicios guiados

1. Debe usar una “ventana privada” en su navegador si desea:

Para navegar completamente anónimo en Internet	
Para no dejar rastro en la computadora que está utilizando	X
Para activar TLS y evitar el seguimiento de cookies	
Para usar DNT	
Para utilizar la criptografía durante la transmisión de datos	

2. ¿Qué es OpenStack?

Un proyecto que permite la creación de IaaS privadas	X
Un proyecto que permite la creación de PaaS privadas	
Un proyecto que permite la creación de SaaS privadas	
Un hipervisor	
Un administrador de contraseñas de código abierto	

3. ¿Cuál de las siguientes opciones son softwares válidos de cifrado de disco ?

RevealJS, EncFS y dm-crypt	
dm-crypt y KeePass	
EncFS y Bitwarden	
EncFS y dm-crypt	X
TLS y dm-crypt	

4. Seleccione verdadero o falso para el cifrado de dispositivo dm-crypt:

Los archivos se cifran antes de escribirse en el disco	T
Todo el sistema de archivos es un blob cifrado	T
Solo los archivos y directorios están encriptados, no los enlaces simbólicos	F
No requiere acceso root	T
Es un cifrado de dispositivo de bloque	T

5. Beamer is:

Un mecanismo de encriptación	
Un hipervisor	
Un software de virtualización	
Un componente OpenStack	
Una herramienta de presentación de LaTeX	X

Respuestas a los ejercicios exploratorios

1. La mayoría de las distribuciones vienen con Firefox instalado de manera predeterminada (si el suyo no lo tiene, primero deberá instalarlo). Vamos a instalar una extensión de Firefox llamada *Lightbeam*, puedes hacerlo presionando `Ctrl` + `Shift` + `A` y buscando "Lightbeam" en el campo de búsqueda que se mostrará en la pestaña abierta, o visitando la página de la extensión con Firefox y haciendo clic en Botón "Instalar": https://addons.mozilla.org/en-US/firefox/addon/lightbeam. Después de hacer esto, inicie la extensión haciendo clic en su icono y comience a visitar algunas páginas web en otras pestañas para ver qué sucede.

   ¿Recuerdas esas cookies que dijimos que pueden compartir tus datos con diferentes servicios cuando visitas un sitio web? Eso es exactamente lo que te mostrará esta extensión. Lightbeam es un experimento de Mozilla que intenta revelar los sitios de primera y tercera parte con los que interactúa al visitar una sola URL. Este contenido generalmente no es visible para el usuario promedio y puede mostrar que a veces un solo sitio web puede interactuar con una docena o más de servicios.

2. ¿Qué es lo más importante cuando se usa un administrador de contraseñas?

   Al usar un administrador de contraseñas, lo más importante es memorizar su contraseña maestra y usar una contraseña aleatoria única para cada servicio diferente.

3. Use su navegador web para navegar a https://haveibeenpwned.com/. Descubra el propósito del sitio web y verifique si su dirección de correo electrónico se incluyó en algunas filtraciones de datos.

   El sitio web mantiene una base de datos de información de inicio de sesión cuyas contraseñas se vieron afectadas por una fuga de contraseña. Este permite buscar una dirección de correo electrónico y muestra si esa dirección de correo electrónico se incluyó en una base de datos pública de credenciales robadas. Lo más probable es que su dirección de correo electrónico haya sido afectada por una u otra fuga. Si ese es el caso, asegúrese de haber actualizado sus contraseñas recientemente. Si aún no utiliza un administrador de contraseñas, eche un vistazo a las recomendadas en esta lección.



Tema 2: Encontrando el camino en un sistema Linux



2.1 Aspectos básicos de la línea de comandos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 2.1

Importancia

3

Áreas de conocimiento clave

- Uso básico de la shell
- Sintaxis de la línea de comandos
- Variables
- Uso de comillas

Lista parcial de archivos, términos y utilidades

- Bash
- `echo`
- `history`
- La variable de entorno `PATH`
- `export`
- `type`



2.1 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objetivo:	2.1 Conceptos básicos de la línea de comandos
Lección:	1 de 2

Introducción

Las distribuciones modernas de Linux tienen una amplia gama de interfaces gráficas de usuario, un administrador siempre necesitará saber cómo trabajar con la línea de comandos o *shell* (como se le llama). Shell es un programa que permite la comunicación basada en texto entre el sistema operativo y el usuario. Por lo general, es un programa en modo de texto que lee la entrada del usuario y la interpreta como comandos para el sistema.

Tenemos diferentes shells en Linux, estos son algunos ejemplos:

- Bourne-again shell (Bash)
- C shell (csh or tcsh, the enhanced csh)
- Korn shell (ksh)
- Z shell (zsh)

En Linux, el más común es el Bash Shell. Este se utilizará en diferentes ejemplos o ejercicios que veremos durante la lectura.

Cuando se utiliza un shell interactivo, el usuario ingresa comandos en un indicador llamado (prompt). En cada distribución de Linux, el indicador predeterminado podría verse un poco diferente, pero generalmente sigue esta estructura:

```
username@hostname current_directory shell_type
```

En Ubuntu o Debian GNU/Linux, la solicitud para un usuario normal probablemente se verá así:

```
carol@mycomputer:~$
```

El indicador del superusuario se verá así:

```
root@mycomputer:~#
```

En CentOS o Red Hat Linux, el mensaje para un usuario normal se vea así:

```
[dave@mycomputer ~]$
```

Y el indicador(prompt) del superusuario se verá así:

```
[root@mycomputer ~]#
```

Vamos a explicar cada componente de la estructura:

`username`
: Nombre del usuario que ejecuta la shell.

`hostname`
: Nombre del host donde se ejecuta el shell. También hay un comando `hostname`, con el cual puede mostrar o configurar el nombre de host del sistema.

`current_directory`
: Es el directorio en el que se encuentra actualmente el shell. El caracter `~` significa que la shell está en el directorio principal (home) del usuario.

`shell_type`

`$` indica que el shell lo ejecuta un usuario normal.

`#` indica que el shell es ejecutado por el superusuario `root`.

Como no necesitamos ningún privilegio especial, utilizaremos una solicitud sin privilegios en los siguientes ejemplos. Por brevedad, solo usaremos el `$` (signo de dólar) como indicador.

Estructura de la línea de comandos

La mayoría de los comandos siguen la misma estructura básica:

```
comando [opción(es)/párametro(s)...]  [argumento(s)...]
```

Tome el siguiente comando como ejemplo:

```
$ ls -l /home
```

Expliquemos el propósito de cada componente:

Comando

Programa que el usuario ejecutó - `ls` en el ejemplo anterior.

Opción(es)/Parámetro(s)

Un “switch” modifica el comportamiento del comando de alguna manera, como es el caso de `-l` (ejemplo anterior). Se puede acceder a las opciones en forma corta y larga. Por ejemplo, `-l` es idéntico a `--format=long`.

También se pueden combinar varias opciones, las letras generalmente se pueden escribir juntas haciendolo de forma abreviada. Por ejemplo, los siguientes comandos hacen lo mismo:

```
$ ls -al
$ ls -a -l
$ ls --all --format=long
```

Argumento(s)

Datos adicionales que requiere el programa, como un nombre de archivo o ruta, como `/home` en el ejemplo anterior.

La única parte obligatoria de esta estructura es el comando. En general, todos los demás elementos son opcionales, pero un programa puede requerir que se especifiquen ciertas opciones, parámetros o argumentos.

> **NOTE** La mayoría de los comandos muestran un breve resumen de las opciones disponibles cuando se ejecutan con el parámetro `--help`. Pronto conoceremos otras formas de aprender más sobre los comandos de Linux.

Tipos de comportamiento de los comando

Shell admite dos tipos de comandos:

Interno

Estos comandos son parte del propio shell y no son programas separados. Hay alrededor de 30 de estos comandos, y su principal objetivo es ejecutar tareas dentro del shell (por ejemplo, `cd`, `set`, `export`).

Externo

Estos comandos residen en archivos individuales; suelen ser programas binarios o scripts. Cuando se ejecuta un comando que no es un shell incorporado, el shell utiliza la variable `PATH` para buscar un archivo ejecutable con el mismo nombre del comando. Además de los programas que se instalan con el administrador de paquetes de la distribución, los usuarios también pueden crear sus propios comandos externos.

El comando `type` muestra de qué tipo es un comando específico:

```
$ type echo
echo is a shell builtin
$ type man
man is /usr/bin/man
```

Comillas (Quoting)

Como usuario de Linux, deberá crear o manipular archivos o variables de diferentes maneras. Esto es fácil cuando se trabaja con nombres de archivo cortos y valores únicos, pero se vuelve más complicado cuando, por ejemplo, hay espacios involucrados, caracteres especiales y variables. Las shells proporcionan una característica llamada comillas que encapsula los datos usando varios tipos de comillas (" ", ' '). En Bash, hay tres tipos de comillas:

- Comillas dobles (Double quotes)

- Comillas simples (Single quotes)
- Caracteres de Escape (Escape characters)

Por ejemplo, los siguientes comandos no actúan de la misma manera debido a las comillas:

```
$ TWOWORDS="two words"
$ touch $TWOWORDS
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol     0 Mar 10 14:56 words
$ touch "$TWOWORDS"
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol     0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  words
$ touch '$TWOWORDS'
$ ls -l
-rw-r--r-- 1 carol carol     0 Mar 10 15:00 '$TWOWORDS'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  two
-rw-r--r-- 1 carol carol     0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol     0 Mar 10 14:56  words
```

NOTE La línea con `TWOWORDS=` es una variable Bash que nosotros mismos creamos. Introduciremos variables más tarde. El objetivo de esta es mostrarle como las comillas afectan el resultado de las variables.

Comillas dobles (Double Quotes)

Las comillas dobles le dicen al shell que interprete el texto entre comillas ("...") como caracteres regulares. Todos los caracteres especiales pierden su significado, excepto el `$` (signo de dólar), `\` (barra invertida) y `` ` `` (comilla invertida). Esto significa que las variables, la sustitución de comandos y las funciones aritméticas todavía se pueden usar.

Por ejemplo, la sustitución de la variable `$USER` no se ve afectada por las comillas dobles:

```
$ echo I am $USER
I am tom
$ echo "I am $USER"
I am tom
```

Por otro lado, un carácter de espacio pierde su significado como separador de argumentos:

```
$ touch new file
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 file
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 new
$ touch "new file"
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:19 new file
```

En el primer ejemplo, el comando `touch` crea dos archivos individuales, el comando interpreta las dos cadenas como argumentos individuales. En el segundo ejemplo, el comando interpreta ambas cadenas como un argumento, por lo tanto, solo crea un archivo. Sin embargo, es una buena práctica evitar el carácter de espacio en los nombres de archivo. En su lugar podría usar un guión bajo (`_`) o un punto (`.`).

Comillas simples (Single Quotes)

Las comillas simples no tienen la excepción de las comillas dobles. Estos revocan cualquier significado especial de cada caracter. Tomemos uno de los primeros ejemplos de arriba:

```
$ echo I am $USER
I am tom
```

Al aplicar las comillas simples, verá un resultado diferente:

```
$ echo 'I am $USER'
I am $USER
```

El comando ahora muestra la cadena exacta sin sustituir la variable.

Caracteres de Escape (Escape characters)

Podemos usar *caracteres de escape* para eliminar significados especiales de los caracteres de Bash. Volvamos a la variable de entorno `$USER`:

```
$ echo $USER
carol
```

Vemos que por defecto, el contenido de la variable se muestra en la terminal. Sin embargo, si precedemos el signo de dólar con un carácter de barra diagonal inversa (\), negará su significado especial. Esto provocara que Bash no expanda el valor de la variable al nombre de usuario de la persona que ejecuta el comando, sino que literalmente interpretará el nombre de la variable:

```
$ echo \$USER
$USER
```

Si recuerdas, podemos obtener resultados similares a estos usando la comilla simple, que imprime el contenido literal de lo que esté entre ellas. Sin embargo, el carácter de escape funciona de manera diferente al indicarle a Bash que ignore cualquier significado especial que pueda tener el personaje que precede

Ejercicios guiados

1. Divida las líneas a continuación en comando, opción(s)/parámetro(s) y argumento(s):

    - Ejemplo: `cat -n /etc/passwd`

Comando:	`cat`
Opción:	`-n`
Argumento:	`/etc/passwd`

    - `ls -l /etc`

Comando:	
Opción:	
Argumento:	

    - `ls -l -a`

Comando:	
Opción:	
Argumento:	

    - `cd /home/user`

Comando:	
Opción:	
Argumento:	

2. Encuentre de que tipo son los siguientes comandos:

    Ejemplo:

`pwd`	Shell builtin
`mv`	External command

`cd`	
`cat`	

`exit`	

3. Resuelva los siguientes comandos que usan comillas:

   Example:

`echo "$HOME is my home directory"`	`echo /home/user is my home directory`

`touch "$USER"`	
`touch 'touch'`	

Ejercicios exploratorios

1. Con un comando y usando brace expansion en Bash (revise la página del manual para Bash), cree 5 archivos numerados del 1 al 5 con el prefijo `game` (`game1`, `game2`, …).

2. Elimine los 5 archivos que acaba de crear con un solo comando, utilizando un carácter especial diferente (revise *Pathname Expansion* en las páginas del manual de Bash).

3. ¿Hay alguna otra forma de hacer que dos comandos interactúen entre sí? ¿Cuáles son?

Resumen

En esta lección, usted aprendió:

- Conceptos del shell de Linux
- ¿Qué es Bash shell?
- La estructura de la línea de comando
- Una introducción a las comillas (quoting)

Comandos utilizados en los ejercicios:

`bash`
: El shell más popular en las computadoras Linux.

`echo`
: Texto de salida en la terminal.

`ls`
: Listar el contenido de un directorio.

`type`
: Muestra cómo se ejecuta un comando específico.

`touch`
: Crea un archivo vacío o actualiza la fecha de modificación de un archivo existente.

`hostname`
: Muestra o cambia el nombre del equipo de un sistema.

Respuestas a los ejercicios guiados

1. Divida las líneas a continuación en comando, opción(s)/parámetro(s) y argumento(s):

   - `ls -l /etc`

Comando:	`ls`
Opción:	`-l`
Argumento:	`/etc`

   - `ls -l -a`

Comando:	`ls`
Opción:	`-l -a`
Argumento:	

   - `cd /home/user`

Comando:	`cd`
Opción:	
Argumento:	`/home/user`

2. Encuentre de que tipo son los siguientes comandos:

`cd`	Shell builtin
`cat`	Comando externo
`exit`	Shell builtin

3. Resuelva los siguientes comandos que usan comillas:

`touch "$USER"`	`tom`
`touch 'touch'`	Creates a file named `touch`

Respuestas a los ejercicios exploratorios

1. Con un comando y usando brace expansion en Bash (revise la página del manual para Bash), cree 5 archivos numerados del 1 al 5 con el prefijo `game` (`game1`, `game2`, …).

   Los rangos se pueden usar para expresar los números del 1 al 5 dentro de un comando:

   ```
   $ touch game{1..5}
   $ ls
   game1  game2  game3  game4  game5
   ```

2. Elimine los 5 archivos que acaba de crear con un solo comando, utilizando un carácter especial diferente (revise *Pathname Expansion* en las páginas del manual de Bash).

   Dado que todos los archivos comienzan con `game` y terminan con un solo carácter (un número del 1 al 5 en este caso), puede usar `?` como un carácter especial para el último carácter en el nombre de archivo:

   ```
   $ rm game?
   ```

3. ¿Hay alguna otra forma de hacer que dos comandos interactúen entre sí? ¿Cuales son?

   Sí, un comando podría escribir datos en un archivo para luego ser procesado por otro. Linux también puede recopilar la salida de un comando y usarlo como entrada para otro comando. Esto se llama *piping* y aprenderemos más sobre esto en una lección futura.



2.1 Lección 2

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objetivo:	2.1 Conceptos básicos de la línea de comandos
Lección:	2 de 2

Introducción

Todas las shells administran información del estado de sus sesiones. La información del tiempo de ejecución(runtime) puede cambiar durante la sesión e influir en el comportamiento del shell. Estos datos también son utilizados por los programas para determinar aspectos de la configuración del sistema. La mayoría de estos datos se almacenan en las denominadas *variables*, las cuales cubriremos en esta lección.

Variables

Las variables son elementos de almacenamiento de datos, como texto o números. Una vez establecida, se puede acceder al valor de una variable en un momento posterior. Las variables tienen un nombre que permite acceder a un dato específico, incluso cuando el contenido de la variable cambia. Son una herramienta muy común en la mayoría de los lenguajes de programación.

En la mayoría de las shells de Linux, hay dos tipos de variables:

Variables locales

Estas variables están disponibles solo para el proceso de shell actual. Si crea una variable local y

luego inicia otro programa desde la misma shell, la variable ya no estará disponible para ese programa, debido a que no son heredadas por subprocesos. Este tipo de variables se denominan variables locales.

Variables de entorno (Environment)

Estas variables están disponibles tanto en una sesión de shell específica como en subprocesos generados a partir de esa misma sesión. Estas variables se pueden usar para pasar datos de configuración a comandos cuando se ejecuten. Debido a que estos programas pueden acceder a estas variables, se denominan variables de entorno. La mayoría de estas variables están en mayúsculas (por ejemplo, `PATH`, `DATE`, `USER`). Algunas variables de entorno proporcionan información como el directorio de inicio del usuario o el tipo de terminal. En ocasiones, el conjunto completo de todas las variables de entorno se conoce como *environment*.

Estos tipos de variables también se conocen como *variables de ámbito* (variable scope).

NOTE Las variables no son persistentes. Cuando se cierra la shell donde se establecieron, se pierden todas las variables y su contenido. La mayoría de las shells proporcionan archivos de configuración que contienen variables que se establecen cada vez que se inicia una nueva shell. Las variables que deben establecerse permanentemente, deben agregarse a uno de estos archivos de configuración.

Manipulando Variables

Como administrador del sistema, deberá crear, modificar o eliminar variables locales y de entorno.

Trabajando con variables locales

Puede configurar una variable local utilizando el operador `=` (igual). Está asignación simple creará una variable local:

```
$ greeting=hello
```

NOTE No agregue ningún espacio antes o después del operador `=`.

Puede mostrar cualquier variable usando el comando `echo`. El comando generalmente muestra el texto en la sección de argumentos:

```
$ echo greeting
greeting
```

Para acceder al valor de la variable, deberá usar `$` (signo de dólar) antes del nombre de la variable.

```
$ echo $greeting
hello
```

Como puede ver, la variable ha sido creada. Ahora abra otro shell e intente mostrar el contenido de esa variable.

```
$ echo $greeting
```

No se muestra nada. Esto significa que las variables siempre existen en una sola shell específica.

Para verificar que la variable es en realidad una variable local, intente generar un nuevo proceso y verifique si este proceso puede acceder a esa variable. Podemos hacerlo iniciando otra shell y dejar que este ejecute el comando `echo`. Como la nueva shell se ejecuta en un nuevo proceso, no heredará variables locales de su proceso padre:

```
$ echo $greeting world
hello world
$ bash -c 'echo $greeting world'
world
```

NOTE Como en el ejemplo anterior, asegúrese de usar comillas simples.

Para eliminar una variable, deberá usar el comando `unset`:

```
$ echo $greeting
hey
$ unset greeting
$ echo $greeting
```

NOTE

`unset` requiere el nombre de la variable como argumento. Por lo tanto, no puede agregar `$` al nombre, ya que pasaría el valor de la variable a `unset` en lugar del nombre de la variable.

Trabajando con variables globales

Para que una variable esté disponible para los subprocesos, tiene que convertir una variable local a una variable de entorno. Esto se hace con el comando `export`. Cuando se invoca con el nombre de la variable, esta se agrega al entorno de shell:

```
$ greeting=hello
$ export greeting
```

NOTE

Asegúrese de no usar `$` cuando ejecute `export`, ya que lo que se debe pasar es el nombre de la variable y no su contenido.

Una forma más fácil de crear la variable de entorno es combinar ambos métodos anteriores, asignando el valor de la variable en el argumento del comando.

```
$ export greeting=hey
```

Verifiquemos nuevamente si la variable es accesible para los subprocesos:

```
$ export greeting=hey
$ echo $greeting world
hey world
$ bash -c 'echo $greeting world'
hey world
```

Otra forma de usar variables de entorno es mediante los comandos. Podemos probar esto con la variable de entorno `TZ` que contiene la zona horaria. El comando `date` usa esta variable para determinar la hora de la zona horaria que se mostrará:

```
$ TZ=EST date
Thu 31 Jan 10:07:35 EST 2019
$ TZ=GMT date
Thu 31 Jan 15:07:35 GMT 2019
```

Puede mostrar todas las variables de entorno utilizando el comando `env`.

La variable `PATH`

La variable `PATH` es una de las variables de entorno más importantes en un sistema Linux. Almacena una lista de directorios separados por dos puntos que contienen programas ejecutables elegibles como comandos de la shell de Linux.

```
$ echo $PATH
/home/user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/ga
mes
```

Para agregar un nuevo directorio a la variable deberá usar el signo de dos puntos (`:`).

```
$ PATH=$PATH:new_directory
```

Aquí un ejemplo:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ PATH=$PATH:/home/user/bin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin
```

Como puede observar, `$PATH` se ve como nuevo valor asignado a `PATH`. Esta variable se resuelve durante la ejecución del comando y se asegura de que se conserve el contenido original de la variable. Por supuesto, en la asignación también puede usar otras variables:

```
$ mybin=/opt/bin
$ PATH=$PATH:$mybin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin:/opt/bi
n
```

La variable `PATH` debe manejarse con precaución, ya que es crucial para trabajar en línea de comandos. Consideremos la siguiente variable `PATH`:


```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Para descubrir cómo el shell invoca un comando específico, puede utilizar el comando `which` con el nombre del comando como argumento. Por ejemplo, podemos tratar de averiguar dónde se almacena `nano`:

```
$ which nano
/usr/bin/nano
```

Como podemos observar, el ejecutable `nano` se encuentra dentro del directorio `/usr/bin`. Eliminemos el directorio de la variable y verifiquemos si el comando aún funciona:

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
```

Busquemos el comando `nano` nuevamente:

```
$ which nano
which: no nano in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games)
```

Como podemos observar, el comando no se encuentra, por lo tanto, no se ejecuta. El mensaje de error también explica la razón por la cual no se encontró y en qué ubicaciones se buscó.

Volvamos a agregar los directorios e intentemos ejecutar el comando nuevamente.

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which nano
/usr/bin/nano
```

Ahora nuestro comando funciona de nuevo.

TIP El orden de los elementos en `PATH` también define el orden de búsqueda. Se ejecuta el primer ejecutable coincidente encontrado al recorrer las rutas.

Ejercicios guiados

1. Cree una variable local `number`.

2. Cree una variable de entorno `ORDER`, utilizando uno de los métodos anteriores.

3. Muestre los nombres de las variables y sus contenidos.

4. ¿Cuáles son los ámbitos de las variables creadas previamente?

Ejercicios exploratorios

1. Cree una variable local `nr_files` y asignele el número de líneas encontradas en el archivo `/etc/passwd`. Sugerencia: mire el comando `wc` y la sustitución de comandos, tampoco se olvide de las comillas.

2. Crea una variable de entorno `ME`. Asigna el valor de la variable `USER` a la misma.

3. Agregue el valor de la variable `HOME` a `ME` usando el delimitador `:`. Luego muestre el contenido de la variable `ME`.

4. Usando el ejemplo de fecha anterior, cree una variable llamada `today` y asignele la fecha para una de las zonas horarias.

5. Cree otra variable llamada `today1` y asígnele la fecha del sistema.

Resumen

En esta lección, usted aprendió:

- Tipos de variables
- Crear variables
- Manipular variables

Comandos utilizados en los ejercicios:

`env`
: Muestra las variables de entorno actual.

`echo`
: Salida de texto.

`export`
: Hace que las variables locales estén disponibles para los subprocesos.

`unset`
: Elimina una variable.

Respuestas a los ejercicios guiados

1. Cree una variable local `number`.

```
$ number=5
```

2. Cree una variable de entorno `ORDER`, utilizando uno de los métodos anteriores.

```
$ export ORDER=desc
```

3. Muestre los nombres de las variables y sus contenidos.

```
$ echo number
number
$ echo ORDER
ORDER
$ echo $number
5
$ echo $ORDER
desc
```

4. ¿Cuáles son los ámbitos de las variables creadas previamente?

   - El alcance de la variable `number` es la subshell actual.
   - El alcance de la variable `ORDER` es el sistema actual.

Respuestas a los ejercicios exploratorios

1. Cree una variable local `nr_files` y asignele el número de líneas encontradas en el archivo `/etc/passwd`. Sugerencia: mire el comando `wc` y la sustitución de comandos, tampoco se olvide de las comillas.

   ```
   $ nr_files=`wc -l /etc/passwd`
   ```

2. Crea una variable de entorno `ME`. Asigna el valor de la variable `USER` a la misma.

   ```
   $ export ME=$USER
   ```

3. Agregue el valor de la variable `HOME` a `ME` usando el delimitador `:`. Luego muestre el contenido de la variable `ME`.

   ```
   $ ME=$ME:$HOME
   $ echo $ME
   user:/home/user
   ```

4. Utilizando el ejemplo de la fecha anterior, cree una variable llamada `today` y asigne la fecha de una de las zonas horarias.

   Se utilizan las zonas horarias GMT y EST como ejemplo, pero cualquier selección de zona horaria es válida.

   ```
   $ today=$(TZ=GMT date)
   $ echo $today
   Thu 31 Jan 15:07:35 GMT 2019
   ```

   or

   ```
   $ today=$(TZ=EST date)
   $ echo $today
   Thu 31 Jan 10:07:35 EST 2019
   ```

5. Cree otra variable llamada `today1` y asígnele la fecha del sistema.

Asumiendo que estás en GMT:

```
$ today1=$(date)
$ echo $today1
Thu 31 Jan 10:07:35 EST 2019
```



2.2 Uso de la línea de comandos para obtener ayuda

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 2.2

Importancia

2

Áreas de conocimiento clave

- Páginas de man
- Páginas de info

Lista parcial de archivos, términos y utilidades

- `man`
- `info`
- `/usr/share/doc/`
- `locate`



2.2 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objectivo:	2.2 Usando la línea de comandos para obtener ayuda
Lección:	1 de 1

Introducción

La línea de comandos es una herramienta muy compleja. Cada comando tiene sus propias opciones únicas, por lo tanto, la documentación es clave cuando se trabaja con un sistema Linux. Además del directorio `/usr/share/doc/` que almacena la mayor parte de la documentación, existen otras herramientas también proporcionan información sobre el uso de comandos. Este capítulo se centra en los métodos para acceder a esa documentación, con el fin de obtener ayuda.

Hay una multitud de métodos para obtener ayuda dentro de la línea de comandos. `man`, `help` e `info` son solo algunos de ellos. Para Linux Essentials, nos centraremos en `man` e `info`, ya que son las herramientas más utilizadas para obtener ayuda.

Otro tema de este capítulo será localizar archivos. Trabajaremos principalmente con el comando `locate`.

Obteniendo ayuda por línea de comandos

Ayuda incorporada

Cuando se comienza con el parámetro `--help`, la mayoría de los comandos muestran algunas instrucciones breves sobre su uso. Aunque no todos los comandos proporcionan está opción, es un buen primer intento para aprender más sobre sus parámetros. Tenga en cuenta que las instrucciones de `--help` a menudo son muy breves en comparación con las otras fuentes de documentación que discutiremos en el resto de esta lección.

Páginas Man

La mayoría de los comandos proporcionan una página de manual o una página "man". Esta documentación generalmente se instala con el software y se puede acceder con el comando `man`. Cuando deseé ver la página del manual de un comando, debe agregar el comando `man` como argumento:

```
$ man mkdir
```

Este comando abre la página del manual para `mkdir`. Puede usar las teclas de flecha arriba y abajo o la barra espaciadora para navegar por la página del manual. Para salir de la página del manual, presione `Q`.

Cada página de `man` está dividida en un máximo de 11 secciones, aunque muchas de estas secciones son opcionales:

Sección	Descripción
NAME	Nombre del comando y breve descripción
SYNOPSIS	Descripción de la sintaxis del comando
DESCRIPTION	Descripción de los efectos del comando
OPTIONS	Opciones disponibles
ARGUMENTS	Argumentos disponibles
FILES	Archivos auxiliares
EXAMPLES	Una muestra de la línea de comando
SEE ALSO	Referencias cruzadas a los temas relacionados
DIAGNOSTICS	Mensajes de advertencia y error
COPYRIGHT	Autor (es) del comando
BUGS	Cualquier limitación conocida del comando

En la práctica, la mayoría de las páginas man no contienen todas estas partes.

Las páginas man están organizadas en ocho categorías, numeradas del 1 al 8:

Categoría	Descripción
1	Comando del usuario
2	Llamadas del sistema
3	Funciones de la biblioteca C
4	Controladores y archivos de dispositivo
5	Archivos de configuración y formatos de archivo
6	Juegos
7	Miscellaneous
8	Comandos del administrador del sistema
9	Funciones del núcleo (no estándar)

Cada página de manual pertenece exactamente a una categoría. Sin embargo, varias categorías pueden contener páginas de manual con el mismo nombre. Tomemos como ejemplo el comando `passwd`. Este comando puede ser usado para cambiar la contraseña de un usuario. Como `passwd` es un comando de usuario, su página man reside en la categoría 1. Además del comando `passwd`, el archivo de base de datos de contraseñas `/etc/passwd` también tiene una página man que se llama `passwd`. Como este archivo es un archivo de configuración, pertenece a la categoría 5. Cuando se hace referencia a una página de manual, a menudo se añade la categoría al nombre de la página de manual, como en `passwd(1)` o `passwd(5)` para identificar la página de manual respectiva.

Por defecto, `man passwd` muestra la primera página man disponible, en este caso `passwd(1)`. La categoría de la página man deseada se puede especificar en un comando como `man 1 passwd` o `man 5 passwd`.

Ya hemos visto cómo navegar a través de una página man y cómo volver a la línea de comando. Internamente, `man` usa el comando `less` para mostrar el contenido de la página. `less` le permite buscar texto dentro de una página man. Para encontrar la palabra `linux` puede usar `/linux` para buscar hacia adelante desde el punto en el que se encuentra en la página o `?Linux` para iniciar una búsqueda hacia atrás. Esta acción resalta todos los resultados coincidentes y mueve la página a la primera coincidencia resaltada. En ambos casos, puede escribir `N` para saltar a la siguiente coincidencia. Para encontrar más información sobre estas características adicionales, presione `H` y se mostrará un menú con toda la información.

Páginas de información (Info Pages)

Otra herramienta que le ayudará mientras trabaja con el sistema Linux son las páginas de información(info pages). Las páginas info suelen ser más detalladas que las páginas de manual y están formateadas en hipertexto, similar a las páginas web en Internet.

Las páginas info se pueden mostrar así:

```
$ info mkdir
```

Para cada página de información, `info` lee un archivo que está estructurado en nodos individuales dentro de un árbol. Cada nodo contiene un tema simple y el comando `info` contiene hipervínculos que pueden ayudarlo a moverse de uno a otro. Puede acceder al enlace presionando "enter" mientras coloca el cursor en uno de los asteriscos principales.

Similar a `man`, la herramienta `info` también tiene comandos de navegación de página. Puede obtener más información sobre estos comandos presionando `?` mientras se encuentra en la página de info. Estas herramientas lo ayudarán a navegar por la página más fácilmente, así como a comprender cómo acceder a los nodos y moverse dentro del árbol.

El directorio `/usr/share/doc/`

Como se mencionó anteriormente, el directorio `/usr/share/doc/` almacena la mayoría de la documentación de cada comando que utilice el sistema. Este directorio contiene una carpeta para la mayoría de los paquetes instalados; el nombre del directorio suele ser el nombre del paquete y en ocasiones incluye su versión. Estos directorios incluyen un archivo `README` o `readme.txt` que contiene la documentación básica del paquete, junto con el archivo `README`. La carpeta también puede contener otros archivos de documentación, como el registro de cambios que incluye el historial del programa en detalle o ejemplos de archivos de configuración para el paquete específico.

La información dentro del archivo `README` varía de un paquete a otro. Todos los archivos están escritos en texto plano, por lo tanto, se pueden leer con cualquier editor de texto preferido. El número exacto y los tipos de archivos dependen del paquete. Consulte algunos de los directorios para obtener una descripción general de sus contenidos.

Localizando archivos

El comando `locate`

Un sistema Linux está construido a partir de numerosos directorios y archivos. Linux tiene muchas herramientas para ubicar un archivo en particular dentro de un sistema. El más rápido es el comando

`locate`.

`locate` busca dentro de una base de datos y luego genera cada nombre que coincida con la cadena brindada:

```
$ locate note
/lib/udev/keymaps/zepto-znote
/usr/bin/zipnote
/usr/share/doc/initramfs-tools/maintainer-notes.html
/usr/share/man/man1/zipnote.1.gz
```

El comando `locate` también admite el uso de comodines y expresiones regulares, por lo tanto, la cadena de búsqueda no tiene que coincidir con el nombre completo del archivo deseado. Aprenderá más sobre las expresiones regulares en un capítulo posterior.

Por defecto, `locate` se comporta como si el patrón estuviera rodeado de asteriscos, por lo que `locate PATTERN` es lo mismo que `locate *PATTERN*`. Esto le permite simplemente proporcionar subcadenas en lugar del nombre del archivo exacto. Puede modificar este comportamiento con las diferentes opciones que puede encontrar explicadas en la página del comando man `locate`.

Debido a que `locate` está leyendo desde una base de datos, es posible que no encuentre un archivo que haya creado recientemente. La base de datos es administrada por un programa llamado `updatedb`. Por lo general, se ejecuta periódicamente, pero si tiene privilegios de root y necesita que la base de datos se actualice de inmediato, puede ejecutar el comando `updatedb` en cualquier momento.

El comando `find`

`find` es otra herramienta muy popular que se utiliza para buscar archivos. Este comando tiene un enfoque diferente, en comparación con el comando `locate`. El comando `find` busca un árbol de directorios de forma recursiva, incluidos sus subdirectorios. `find` realiza dicha búsqueda en cada invocación y no mantiene una base de datos como `locate`. Al igual que `locate`, `find` también admite comodines y expresiones regulares.

`find` requiere al menos la ruta que debe buscar. Además, se pueden agregar las llamadas expresiones para proporcionar criterios de filtro para indicar que archivos mostrar. Un ejemplo es la expresión `-name` que busca archivos con un nombre específico:

```
~$ cd Downloads
~/Downloads
$ find . -name thesis.pdf
./thesis.pdf
~/Downloads
$ find ~ -name thesis.pdf
/home/carol/Downloads/thesis.pdf
```

El primer comando `find` busca el archivo dentro del directorio actual de `Downloads`, mientras que el segundo busca el archivo en el directorio de inicio del usuario.

El comando `find` es muy complejo, por lo tanto no será cubierto en el examen de Linux Essentials. Sin embargo, es una herramienta poderosa que es particularmente útil en la práctica.

Ejercicios guiados

1. Use el comando `man` para averiguar qué hace cada comando

Comando	Descripción
`ls`	Mostrar el contenido de un directorio.
`cat`	
`cut`	
`cd`	
`cp`	
`mv`	
`mkdir`	
`touch`	
`wc`	
`passwd`	
`rm`	
`rmdir`	
`more`	
`less`	
`whereis`	
`head`	
`tail`	
`sort`	
`tr`	
`chmod`	
`grep`	

2. Abra la página de información `ls` e identifique el MENU.
   - ¿Qué opciones tienes?
   - Encuentre la opción que le permite ordenar la salida por tiempo de modificación.

3. Muestre la ruta de los primeros 3 archivos `README`. Use el comando `man` para identificar la opción

correcta para `locate`.

4. Cree un archivo llamado `test` en su directorio de inicio. Encuentre su ruta absoluta con el comando `locate`.

5. ¿Lo encontraste de inmediato? ¿Qué tuvo que hacer para que `locate` lo encontrara?

6. Busque el archivo de prueba que creó anteriormente, utilizando el comando `find`. ¿Qué sintaxis uso y cuál es la ruta absoluta?

Ejercicios exploratorios

1. Hay un comando en la tabla anterior que no tiene una página `man`. ¿Cuál es y por qué cree que el comando no tiene una página de manual?

2. Usando los comandos de la tabla anterior, cree el siguiente árbol de archivos. Los nombres que comienzan con mayúscula son directorios y los que están en minúsculas son archivos.

```
User
□□□ Documents
|   □□□Hello
|   |   □□□hey2
|   |   □□□helloa
|   |   □□□ola5
|   □□□World
|       □□□earth9
□□□ Downloads
|   □□□Music
|   □□□Songs
|       □□□collection1
|       □□□collection2
□□□ Test
|   □□□ passa
□□□ test
```

3. Muestre en la pantalla el directorio de trabajo actual, incluidas las subcarpetas.

4. Busque dentro del árbol todos los archivos que terminen con un número.

5. Elimine todo el árbol de directorios con un solo comando.

Resumen

En esta lección usted aprendió:

- ¿Cómo obtener ayuda?
- Usar el comando `man`
- Navegar por la página `man`
- Diferentes secciones de la página `man`
- Usar el comando `info`
- Navegar entre diferentes nodos
- Buscar archivos dentro del sistema

Comandos utilizados en los ejercicios:

`man`
: Muestra una página de manual.

`info`
: Muestra una página de información.

`locate`
: Busca en la base de datos `locate` archivos con un nombre específico.

`find`
: Busca en el sistema de archivos nombres que coincidan con un conjunto de criterios seleccionados.

`updatedb`
: Actualiza la base de datos `locate`,

Respuestas a los ejercicios guiados

1. Use el comando `man` para averiguar qué hace cada comando

Comando	Descripción
`ls`	Muestra el contenido de un directorio.
`cat`	Concatena o ve archivos de texto.
`cut`	Elimina secciones de un archivo de texto.
`cd`	Cambia a un directorio diferente.
`cp`	Copia un archivo.
`mv`	Mueve un archivo (también se puede usar para cambiar el nombre).
`mkdir`	Crea un nuevo directorio.
`touch`	Crea un archivo o modifica la fecha y hora de la última modificación de un archivo existente.
`wc`	Cuenta el número de palabras, líneas o bytes de un archivo.
`passwd`	Cambia la contraseña de un usuario.
`rm`	Elimina un archivo.
`rmdir`	Elimina un directorio.
`more`	Visualiza archivos de texto una pantalla a la vez.
`less`	Visualiza archivos de texto, permite desplazarse hacia arriba y hacia abajo una línea o página a la vez.
`whereis`	Muestra la ruta del archivo de un programa específico y de manuales relacionados.
`head`	Muestra las primeras líneas de un archivo.
`tail`	Muestra las últimas líneas de un archivo.
`sort`	Ordena un archivo de manera numérica o alfabéticamente.
`tr`	Traduce o elimina caracteres de un archivo.
`chmod`	Cambia los permisos de un archivo.

Comando	Descripción
`grep`	Búsquedas dentro de un archivo.

2. Abra la página de información `ls` e identifique el MENU.
   - ¿Qué opciones tienes?
     - ¿Qué archivos están listados?
     - ¿Qué información está listada?
     - Ordenar la salida
     - Detalles sobre el ordenamiento de versiones
     - Formato de salida general
     - Formateando marcas de tiempo de archivo
     - Formateando los nombres de archivo
   - Encuentre la opción que le permite ordenar la salida por tiempo de modificación.

     `-t` or `--sort=time`

3. Muestre la ruta a los primeros 3 archivos `README`. Use el comando `man` para identificar la opción correcta para `locate`.

```
$ locate -l 3 README
/etc/alternatives/README
/etc/init.d/README
/etc/rc0.d/README
```

4. Cree un archivo llamado `test` en su directorio de inicio. Encuentre su ruta absoluta con el comando `locate`.

```
$ touch test
$ locate test
/home/user/test
```

5. ¿Lo encontraste de inmediato? ¿Qué tuvo que hacer para que 'locate' lo encontrara?

```
$ sudo updatedb
```

The file is newly created, therefore there is no record of it in the database.

6. Busque el archivo de prueba que creó anteriormente, utilizando el comando `find`. ¿Qué sintaxis uso y cuál es la ruta absoluta?

```
$ find ~ -name test
```

o

```
$ find . -name test
/home/user/test
```

Respuestas a los ejercicios exploratorios

1. Hay un comando en la tabla anterior que no tiene una página `man`. ¿Cuál es y por qué cree que el comando no tiene una página de manual?

   El comando `cd`. No tiene una página de manual porque es un comando interno de shell.

2. Usando los comandos de la tabla anterior, cree el siguiente árbol de archivos. Los nombres que comienzan con mayúscula son directorios y los que están en minúsculas son archivos.

```
User
□□□ Documents
|    □□□Hello
|    |    □□□hey2
|    |    □□□helloa
|    |    □□□ola5
|    □□□World
|         □□□earth9
□□□ Downloads
|    □□□Music
|    □□□Songs
|         □□□collection1
|         □□□collection2
□□□ Test
|    □□□ passa
□□□ test
```

   La solución es una combinación de comandos `mkdir` y `touch`.

3. Muestre en la pantalla el directorio de trabajo actual, incluidas las subcarpetas.

```
$ ls -R
```

4. Search within the tree for all files that end with a number.

```
$ find ~ -name "*[0-9]"
$ locate "*[0-9]"
```

5. Elimine todo el árbol de directorios con un solo comando.

```
$ rm -r Documents Downloads Test test
```



2.3 Uso de directorios y listado de archivos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 2.3

Importancia

2

Áreas de conocimiento clave

- Archivos y directorios
- Archivos ocultos y directorios
- Directorios home
- Rutas absolutas y relativas

Lista parcial de archivos, términos y utilidades

- Opciones comunes para el comando `ls`
- Listados recursivos
- `cd`
- `.` y `..`
- `home` y `~`



2.3 Lección 1

Introducción

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objectivo:	2.3 Usando directorios y listando archivos
Lección:	1 de 2

Archivos y Directorios

El sistema de archivos Linux es similar a otros sistemas de archivos de otros sistemas operativos, ya que contiene *ficheros* y *directorios*. Los ficheros almacenan datos como texto (legible por el ser humano), programas ejecutables o datos binarios que son interpretados por el computador. Mientras que los directorios se utilizan para crear una organización dentro del sistema de archivos. Los directorios pueden contener archivos y otros directorios.

```
$ tree

Documents
��� Mission-Statement.txt
��� Reports
    ��� report2018.txt

1 directory, 2 files
```

En este ejemplo, `Documents` es un directorio que contiene un archivo (`Mission-Statement.txt`) y un *subdirectorio* (`Reports`) El directorio `Reports` a su vez contiene un archivo llamado `report2018.txt`. El directorio `Documents` se dice que es el *padre* del directorio `Reports`.

> **TIP** Si el comando `tree` no está disponible en su sistema, instálelo usando el gestor de paquetes de su distribución Linux. Consulte la lección sobre gestión de paquetes para aprender a como realizarlo.

Nombres de archivos y directorios

Los nombres de archivos y directorios en Linux pueden contener letras minúsculas y mayúsculas, números, espacios y caracteres especiales; sin embargo, dado que muchos caracteres especiales tienen un significado diferente en el intérprete de comandos de Linux, es una buena práctica no utilizar espacios o caracteres especiales cuando se nombran archivos o directorios. Un ejemplo de esto son los espacios, ya que necesitan que el carácter escape (*escape character*) `\` sea introducido correctamente:

```
$ cd Mission\ Statements
```

También observemos el nombre del archivo `report2018.txt`. Los nombres de archivos pueden contener un *sufijo* que se encuentra después del punto (`.`). A diferencia de Windows, este sufijo no tiene un significado especial en Linux; solo está ahí para la comprensión humana. En nuestro ejemplo, el `.txt` nos indica que se trata de un archivo de texto plano, aunque técnicamente podría contener cualquier tipo de dato.

Navegando en el Sistema de Archivos

Obteniendo la ubicación actual

Dado que los shells de Linux como Bash están basados en texto, es importante recordar la ubicación actual cuando se navega por el sistema de archivos, el *command prompt* puede proporcionar esta información:

```
user@hostname ~/Documents/Reports $
```

Tenga en cuenta que información como `usuario` y `nombre de host` se cubrirá en futuras secciones. Desde el prompt, ahora sabemos que nuestra ubicación actual se encuentra en el directorio `Reports`. De forma similar, el comando `pwd` imprimirá el directorio de trabajo *(working directory)*:

```
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
```

La relación de directorios se representa con una barra oblicua (/). Sabemos que `Reports` es un subdirectorio de `Documents` y este es un subdirectorio de `user`, que se encuentra en un directorio llamado `home`. El directorio `home` no parece tener un directorio padre, pero esto no es cierto en lo absoluto; el padre de `home` se llama *root*, y está representado por la primera barra oblicua (/) que discutiremos en una sección posterior.

Note que la salida del comando `pwd` difiere ligeramente de la ruta dada por el command prompt. En lugar de `/home/user` contiene una tilde (~). La tilde es un carácter especial que representa el directorio home del usuario. Esto será cubierto con más detalle en la siguiente lección.

Listando el contenido de los directorios

El contenido del directorio actual se lista con el comando `ls`:

```
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Tenga en cuenta que `ls` no proporciona información sobre el directorio padre. Por defecto `ls` no muestra ninguna información sobre el contenido de los subdirectorios. El comando `ls` sólo puede "ver" lo que hay en el directorio actual.

Cambiando al directorio actual

La navegación en Linux se hace principalmente con el comando `cd`. Esto *cambia el directorio*. Usando el comando `pwd` de antes, sabemos que nuestro directorio actual es `/home/user/Documents/Reports`. Podemos cambiar nuestro directorio actual introduciendo una nueva ruta:

```
user@hostname ~ $ cd /home/user/Documents
user@hostname ~/Documents $ pwd
/home/user/Documents
user@hostname ~/Documents $ ls
Mission-Statement.txt Reports
```

Desde nuestra nueva ubicación, podemos "ver" `Mission-Statement.txt` y nuestro subdirectorio `Reports`, pero no el contenido de nuestro subdirectorio:

```
user@hostname ~/Documents $ cd Reports
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Ahora estamos de vuelta donde empezamos.

Rutas Absolutas y Relativas

El comando `pwd` siempre imprime una ruta *absoluta*. Esto significa que la ruta contiene cada paso de la ruta, desde la parte superior del sistema de ficheros (`/`) hasta la parte inferior (`Reports`). Las rutas absolutas siempre comienzan con un `/`.

```
/
└── home
    └── user
        └── Documents
            └── Reports
```

La ruta absoluta contiene toda la información necesaria para llegar a `Reports` desde cualquier parte del sistema de ficheros, el inconveniente es que es tedioso de escribir.

El segundo ejemplo (`cd Reports`) era mucho más fácil de escribir, este es un ejemplo de una ruta *relativa*. Las rutas relativas son más cortas, pero sólo tienen significado en relación a tu ubicación actual. Considera esta analogía: te estoy visitando en tu casa y me dices que tu amigo vive en la casa de al lado, entenderé esa ubicación porque es relativa a mi ubicación actual, pero si me dices esto por teléfono, no podré encontrar la casa de tu amigo, necesitarás darme la dirección completa de la calle.

Rutas Relativas Especiales

Para revelar las primeras rutas especiales, entramos en el comando `ls` con la bandera `-a`. Esta bandera modifica el comando `ls` para que se listen todos los archivos y directorios *all*, incluyendo los archivos y directorios ocultos:

```
user@hostname ~/Documents/Reports $ ls -a
.
..
report2018.txt
```

NOTE Puedes consultar la página de `man` para `ls` entender lo que `-a` está haciendo aquí.

Este comando ha revelado dos resultados adicionales: Estas son rutas especiales, no representan nuevos archivos o directorios, sino que representan directorios que usted ya conoce:

`.`
: Indica la *ubicación actual* (en este caso, `Reports`).

`..`
: Indica el *directorio padre* (en este caso, `Documents`).

Por lo general, no es necesario usar una ruta relativa especial para la ubicación actual. Es más fácil y comprensible escribir `report2018.txt` que `./report2018.txt`, pero el `.` tiene usos que aprenderá en futuras secciones. Por ahora, nos centraremos en la ruta relativa para el directorio padre:

```
user@hostname ~/Documents/Reports $ cd ..
user@hostname ~/Documents $ pwd
/home/user/Documents
```

El ejemplo de `cd` es mucho más fácil cuando se usa `..` en lugar de la ruta absoluta, además, podemos combinar este patrón para navegar por el árbol de archivos muy rápidamente.

```
user@hostname ~/Documents $ cd ../..
$ pwd
/home
```

Ejercicios guiados

1. Para cada una de las siguientes rutas, identifique si es *absoluta* o *relativa*:

`/home/user/Downloads`	
`../Reports`	
`/var`	
`docs`	
`/`	

2. Observe la siguiente estructura de archivos. Nota: Los directorios terminan con una barra inclinada (/) cuando se ejecuta `tree` con la opción `-F`. Necesitará privilegios elevados para ejecutarlo en el directorio principal root (/). El siguiente es un resultado de ejemplo y no es indicativo de una estructura de directorio completa. Úselo para responder las siguientes preguntas:

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directorios, 5 archivos
```

Use esta estructura para responder las siguientes preguntas.

Un usuario ingresa los siguientes comandos:

```
$ cd /etc/udev
$ ls -a
```

¿Cuál será la salida del comando `ls -a`?

3. Ingrese el comando más corto posible para cada una de las siguientes situaciones:
   - Su ubicación actual es root (/). Ingrese el comando para navegar a `lost+found` dentro del directorio `home` (ejemplo):

     ```
     $ cd home/lost+found
     ```

   - Su ubicación actual es root (/). Ingrese el comando para navegar al directorio llamado `/etc/network/`.

   - Su ubicación actual es `/home/user/Documents/`. Navegue al directorio llamado `/etc/`.

   - Su ubicación actual es `/etc/systemd/system/`. Navegue al directorio llamado `/home/user/`.

4. Considere los siguientes comandos:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

¿Cuál es el resultado del comando final `pwd`?

Ejercicios exploratorios

1. Supongamos que un usuario ha ingresado los siguientes comandos:

```
$ mkdir "this is a test"
$ ls
this is a test
```

   ¿Qué comando `cd` le permitiría ingresar a este directorio?

2. Intente esto nuevamente, pero después de escribir `cd this`, presione la tecla TAB. ¿Qué se muestra ahora en la solicitud?

   Este es un ejemplo de _ autocompletado_, la cual es una herramienta invaluable no solo para ahorrar tiempo, sino también para evitar errores ortográficos.

3. Intente crear un directorio cuyo nombre contenga un carácter `\`. Visualice el nombre del directorio con `ls` y elimine el directorio.

Resumen

En esta lección usted aprendió:

- Los fundamentos del sistema de archivos Linux
- La diferencia entre *directorios padres* y *subdirectorios*
- La diferencia entre las rutas *absolutas* y *relativas*
- Las rutas relativas especiales `.` y `..`
- Navegar por el sistema de archivos usando `cd`
- Mostrar tu ubicación actual usando `pwd`
- Listar *todos* los archivos y directorios usando `ls -a`

Los siguientes comandos se discutieron en esta lección:

`cd`
: Cambiar el directorio actual.

`pwd`
: Imprime la ruta actual del directorio de trabajo.

`ls`
: Enumerar el contenido de un directorio y mostrar las propiedades de los archivos.

`mkdir`
: Crear un nuevo directorio.

`tree`
: Muestra la jerárquica de un árbol de directorios.

Respuestas a los ejercicios guiados

1. Para cada una de las siguientes rutas, identifique si es *absoluta* o *relativa*:

`/home/user/Downloads`	absoluta
`../Reports`	relativa
`/var`	absoluta
`docs`	relativa
`/`	absoluta

2. Observe la siguiente estructura de archivos. Nota: Los directorios terminan con una barra inclinada (/) cuando se ejecuta `tree` con la opción `-F`. Necesitará privilegios elevados para ejecutarlo en el directorio principal root (/). El siguiente es un resultado de ejemplo y no es indicativo de una estructura de directorio completa. Úselo para responder las siguientes preguntas:

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directorios, 4 archivos
```

Un usuario ingresa los siguientes comandos:

```
$ cd /etc/udev
$ ls -a
```

¿Cuál será la salida del comando `ls -a`?

```
. .. rules.d udev.conf
```

3. Ingrese el comando más corto posible para cada una de las siguientes situaciones:
   - Su ubicación actual es root (/). Ingrese el comando para navegar a `lost+found` dentro del directorio `home` (ejemplo):

     ```
     $ cd home/lost+found
     ```

   - Su ubicación actual es root (/). Ingrese el comando para navegar al directorio llamado `/etc/network/`

     ```
     $ cd etc/network
     ```

   - Su ubicación actual es `/home/user/Documents/`. Navegue al directorio llamado `/etc/`.

     ```
     $ cd /etc
     ```

   - Su ubicación actual es `/etc/systemd/system/`. Navegue al directorio llamado `/home/user/`.

     ```
     $ cd /home/user
     ```

4. Considere los siguientes comandos:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

¿Cuál es el resultado del comando final `pwd`?

```
/etc/systemd
```

Respuestas a los ejercicios exploratorios

1. Supongamos que un usuario ha ingresado los siguientes comandos:

   ```
   $ mkdir "this is a test"
   $ ls
   this is a test
   ```

   ¿Qué comando `cd` le permitiría ingresar a este directorio?

   ```
   $ cd this\ is\ a\ test
   ```

2. Intente esto nuevamente, pero después de escribir `cd this`, presione la tecla TAB. ¿Qué se muestra ahora en la solicitud?

   ```
   $ cd this\ is\ a\ test
   ```

   Este es un ejemplo de _ autocompletado_, que es una herramienta invaluable no solo para ahorrar tiempo, sino también para evitar errores ortográficos.

3. Intente crear un directorio cuyo nombre contenga un carácter `\`. Visualice el nombre del directorio con `ls` y elimine el directorio.

   Puedes escapar de la barra invertida usando otra igual (`\\`) o usar comillas simples o dobles en todo el nombre del directorio:

   ```
   $ mkdir my\\dir
   $ ls
   'my\dir'
   $ rmdir 'my\dir'
   ```



2.3 Lección 2

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objectivo:	2.3 Usando directorios y listando archivos
Lección:	2 de 2

Introducción

El sistema operativo Unix se diseñó originalmente para computadoras mainframe a mediados de la década de 1960. Estas computadoras fueron compartidas entre muchos usuarios quienes accedieron a los recursos del sistema a través de *terminales*. Estas ideas se transmiten hoy en día a los sistemas Linux donde todavía se habla sobre el uso de "terminales" para ingresar comandos en el shell y donde cada sistema Linux está organizado de tal manera que es fácil crear varios usuarios en un solo sistema.

Directorios principales

Este es un ejemplo de un sistema de archivos normal en Linux:


```
$ tree -L 1 /
/
├── bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

La mayoría de estos directorios son consistentes en todos los sistemas Linux desde pequeños sistemas embebidos, servidores y hasta supercomputadoras. Un usuario experimentado de Linux puede estar seguro de que puede encontrar el comando `ls` dentro de `/bin`, puede cambiar la configuración del sistema modificando archivos en `/etc`, y leer los registros del sistema `/var`. La ubicación de estos archivos y directorios está definido por el Estándar de jerarquía del sistema de archivos (FHS), el cual se discutirá en una lección posterior. Aprenderá más sobre el contenido de estos directorios a medida que continúe aprendiendo más sobre Linux, pero por el momento, sepa que:

- Los cambios que realice en el sistema de archivos raíz afectarán a todos los usuarios.
- Cambiar archivos en el sistema de archivos raíz requerirá permisos de administrador.

Esto significa que los usuarios normales tendrán prohibido modificar estos archivos, y también se les puede prohibir incluso leerlos. Cubriremos el tema de los permisos en una sección posterior.

Ahora, nos centraremos en el directorio `/home` que debería ser algo familiar en este punto:

```
$ tree -L 1 /home
/home
▯▯▯ user
▯▯▯ michael
▯▯▯ lara
```

En el ejemplo nuestro sistema tiene tres usuarios normales y cada uno tiene su propia ubicación donde pueden crear y modificar archivos y directorios sin afectar a su vecino. Por ejemplo, en la lección anterior estábamos trabajando con la siguiente estructura de archivos:

```
$ tree /home/user
user
▯▯▯ Documents
    ▯▯▯ Mission-Statement
    ▯▯▯ Reports
        ▯▯▯ report2018.txt
```

En realidad, el sistema de archivos real puede verse así:

```
$ tree /home
/home
▯▯▯ user
▯   ▯▯▯ Documents
▯       ▯▯▯ Mission-Statement
▯       ▯▯▯ Reports
▯           ▯▯▯ report2018.txt
▯▯▯ michael
▯   ▯▯▯ Documents
▯   ▯   ▯▯▯ presentation-for-clients.odp
▯   ▯▯▯ Music
```

…y así sucesivamente para `lara`.

En Linux, `/home` es similar a un edificio de departamentos. Muchos usuarios pueden tener su espacio separados en departamentos dedicados. Las utilidades y el mantenimiento del edificio en sí son responsabilidad del superintendente (usuario root).

La ruta relativa especial para home

Cuando inicia una nueva sesión de terminal en Linux, ve un símbolo del sistema similar a este:


```
user@hostname ~ $
```

La tilde (~) aquí representa nuestro *directorio home* . Si ejecuta el comando `ls` verá algunos resultados familiares:

```
$ cd ~
$ ls
Documents
```

Compare esto con el sistema de archivos anterior para verificar su comprensión.

Considere ahora lo que sabemos sobre Linux: es similar a un edificio de departamentos con muchos usuarios que residen en `/home`. Por lo tanto, el home del `user` será diferente al home del usuario `michael`. Para demostrar esto, usaremos el comando `su` para *cambiar de usuario*.

```
user@hostname ~ $ pwd
/home/user
user@hostname ~ $ su - michael
Password:
michael@hostname ~ $ pwd
/home/michael
```

El significado de `~` cambia dependiendo de quién sea el usuario. Para `michael`, la ruta absoluta de `~` es `/home/michael`. Para `lara`, la ruta absoluta de `~` es `/home/lara`, y así sucesivamente.

Rutas relativas a los archivos de inicio (relative-to-home)

Es muy útil usar `~` siempre que no cambie de usuario. Consideraremos el siguiente ejemplo para `user` que ha comenzado una nueva sesión:

```
$ ls
Documents
$ cd Documents
$ ls
Mission-Statement
Reports
$ cd Reports
$ ls
report2018.txt
$ cd ~
$ ls
Documents
```

Tenga en cuenta que los usuarios siempre comenzarán una nueva sesión en su directorio de inicio (home). En este ejemplo, `user` ha navegado a su subdirectorio `Documents/Reports` y con el comando `cd ~` ha regresado a donde comenzó. Puede realizar la misma acción utilizando el comando `cd` sin argumentos:

```
$ cd Documents/Reports
$ pwd
/home/user/Documents/Reports
$ cd
$ pwd
/home/user
```

Como último punto a tener en cuenta: podemos especificar los directorios de inicio de otros usuarios especificando el nombre después de la tilde. Por ejemplo:

```
$ ls ~michael
Documents
Music
```

Tenga en cuenta que esto solo funcionará si `michael` nos ha dado permiso para ver el contenido de su directorio de inicio.

Consideremos una situación en la que `michael` quisiera ver el archivo `report2018.txt` en el directorio de inicio de `user`. Suponiendo que `michael` tiene permiso para hacerlo, puede usar el comando `less`.

```
$ less ~user/Documents/Reports/report2018.txt
```

Cualquier ruta de archivo que contenga el carácter `~` se llama ruta *relative-to-home*.

Archivos y directorios ocultos

En la lección anterior, presentamos la opción `-a` para el comando `ls`. Utilizamos `ls -a` para introducir a las rutas relativas especiales: `.` y `..`. La opción `-a` enumerará *todos* los archivos y directorios incluidos los *ocultos*.

```
$ ls -a ~
.
..
.bash_history
.bash_logout
.bash-profile
.bashrc
Documents
```

Los archivos y directorios ocultos siempre comenzarán con un punto (`.`). Por defecto, el directorio de inicio(home) de un usuario incluirá muchos archivos ocultos que a menudo se usan para establecer configuraciones específicas del usuario. Estas configuraciones solo deben ser modificadas por un usuario experimentado.

La opción de listado largo (*long list*)

El comando `ls` tiene muchas opciones para cambiar su comportamiento. Revisemos una de las más comunes:

```
$ ls -l
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

El argumento `-l` crea un *listado largo de detalles (long list)*. Cada archivo y directorio ocuparán una línea, pero se mostrará información adicional sobre cada uno de ellos.

`-rw-r-r--`

Tipo de archivo y permisos del archivo. Tenga en cuenta que un archivo normal comenzará con guión y un directorio comenzará con `d`.

`1`
: Número de enlaces al archivo.

`user staff`
: Especifica la propiedad del archivo, `user` es el propietario del archivo el cual también está asociado al grupo `staff`.

`3606`
: Tamaño del archivo en bytes.

`Jan 13 2017`
: Fecha de la última modificación del archivo.

`report2018.txt`
: Nombre del archivo.

Temas como propietarios, permisos y enlaces serán cubiertos en secciones futuras. Como puede observar, a menudo el formato de listado largo de `ls` es preferible que el predeterminado.

Opciones adicionales del comando ls

A continuación, presentamos algunas de las formas más comunes al usar el comando `ls`. Como podrá observar, el usuario puede combinar muchas opciones para obtener el resultado deseado.

`ls -lh`
: La combinación de *long list* con *human readable* nos dará el tamaño de los archivos con sufijos útiles como `M` para megabytes o `K` para kilobytes.

`ls -d */`
: La opción `-d` enumerará los directorios pero no sus contenidos. La combinación con `*/` mostrará solo subdirectorios y ningún archivo.

`ls -lt`
: Combinar *long list* con la opción de ordenar por *modification time*. Los archivos con los cambios más recientes estarán en la parte superior y los archivos con los cambios más antiguos estarán en la parte inferior, pero este orden se puede revertir con:

`ls -lrt`
: Combinar *long list* con *sort by (modification) time*, combinado con `-r` que *invierte* el orden. Ahora los archivos con los cambios más recientes se encuentran al final de la lista. Además de ordenar

por *modification time*, los archivos también se pueden ordenar por *access time* o por *status time*.

`ls -lX`

Combinar *long list* con la opción de ordenar por *file eXtension*. Esto agrupará todos los archivos que terminen con `.txt`, todos los que terminen con `.jpg` y así sucesivamente.

`ls -S`

El `-S` se ordena por archivo *size* de forma muy similar a `-t` y `-X`, tiempo y extensión respectivamente. Los archivos más grandes estarán primero y los más pequeños al final. Tenga en cuenta que el contenido de los subdirectorios no se incluye en el orden.

`ls -R`

La opción `-R` modificará el comando `ls` para mostrar una lista *recursiva*. ¿Qué significa esto?

Recursión en Bash

La recursión se refiere a una situación en la que "algo se define en términos de sí mismo". La recursión es un concepto muy importante en la informática, pero aquí su significado es mucho más simple. Consideremos nuestro ejemplo de antes:

```
$ ls ~
Documents
```

Sabemos desde antes que `user` tiene un directorio personal, y en este directorio hay un subdirectorio. Hasta ahora `ls` sólo nos ha mostrado los ficheros y subdirectorios de una ubicación, pero no puede decirnos el contenido de estos subdirectorios. En estas lecciones, hemos utilizado el comando `tree` cuando queríamos mostrar el contenido de muchos directorios. Desafortunadamente, `tree` no es una de las utilidades principales de Linux y por lo tanto no está siempre disponible. Compare la salida de `tree` con la salida de `ls -R` en los siguientes ejemplos:

```
$ tree /home/user
user
▯▯▯ Documents
    ▯▯▯ Mission-Statement
    ▯▯▯ Reports
        ▯▯▯ report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Como puede observar, con la opción recursiva obtenemos una lista mucho más larga de archivos. De hecho, es como si ejecutáramos el comando `ls` en el directorio de inicio de `user` y encontráramos un subdirectorio, luego entramos en ese subdirectorio y ejecutamos el comando `ls` nuevamente donde encontramos el archivo `Mission-Statement` y otro subdirectorio llamado `Reports`, y nuevamente, ingresamos al subdirectorio y ejecutamos el comando `ls`. Prácticamente, ejecutar `ls -R` es como decirle a Bash: “Ejecute `ls` aquí y repita el comando en cada subdirectorio que encuentre”.

La recursión es particularmente importante en los comandos de modificación de archivos, como copiar o eliminar directorios. Por ejemplo, si desea copiar el subdirectorio `Documents`, necesitara especificar una copia recursiva para extender este comando a todos los subdirectorios.

Ejercicios guiados

1. Use la siguiente estructura de archivos para responder las siguientes tres preguntas:

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

   - ¿Qué comando navegará en el directorio `network` independientemente de su ubicación actual?

   - ¿Qué comando puede ingresar `user` para navegar en su directorio `Documents` desde `/etc/udev`? Use el camino más corto posible.

   - ¿Qué comando puede ingresar `user` para navegar en el directorio `Music` de `michael`? Use el camino más corto posible.

2. Considere la siguiente salida de `ls -lh` para responder las siguientes dos preguntas. Tenga en cuenta que los directorios se indican con una `d` al comienzo de la línea.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-
24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-
07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-
30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

- Cuando ejecutas el comando `ls -lrS`. ¿Qué archivo estará al principio?

- Describa lo que espera ver al ejecutar `ls -ad */`.

Ejercicios exploratorios

1. Ejecute el comando `ls -lh` en un directorio que contenga subdirectorios. Tenga en cuenta el tamaño indicado de estos directorios. ¿Te parecen correctos estos tamaños de archivo? ¿Representan con precisión el contenido de todos los archivos dentro de ese directorio?

2. Aquí hay un nuevo comando para probar: `du -h`. Ejecute este comando y describa el resultado que le proporciona.

3. En muchos sistemas Linux, puedes escribir `ll` y obtener el mismo resultado como si escribieras `ls -l`. Sin embargo, tenga en cuenta que `ll` *no* es un comando. Por ejemplo, `man ll` le dará el mensaje de que no existe un manual para esto. Este es un ejemplo de un *alias*. ¿Por qué los alias podrían ser útiles para un usuario?

Resumen

En esta lección usted aprendió:

- Que cada usuario de Linux tendrá un directorio de inicio.
- Que puede acceder al directorio de inicio del usuario actual utilizando `~`.
- Cualquier ruta de archivo que use `~` se llama ruta *relative-to-home*.

También aprendió sobre algunas de las formas más comunes de modificar el comando `ls`.

`-a` (all)
: Imprime todos los archivos / directorios, incluidos los ocultos.

`-d` (directories)
: Enumera directorios, pero sin sus contenidos.

`-h` (human readable)
: Imprime tamaños de archivo en formato legible para humanos.

`-l` (long list)
: Proporciona detalles adicionales en un archivo/directorio por línea.

`-r` (reverse)
: Invierte el orden.

`-R` (recursive)
: Enumera todos los archivos, incluidos los archivos de cada subdirectorio.

`-S` (size)
: Ordena por tamaño de archivo.

`-t` (time)
: Ordena por fecha de modificación.

`-X` (eXtension)
: Ordena por extensión de archivo.

Respuestas a los ejercicios guiados

1. Use la siguiente estructura de archivos para responder las siguientes tres preguntas:

```
/
├── etc/
│   ├── network/
│   │   ├── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

   - ¿Qué comando navegará en el directorio `network` independientemente de su ubicación actual?

     ```
     cd /etc/network
     ```

   - ¿Qué comando puede ingresar `user` para navegar en su directorio `Documents` desde `/etc/udev`? Use el camino más corto posible.

     ```
     cd ~/Documents
     ```

   - ¿Qué comando puede ingresar `user` para navegar en el directorio `Music` de `michael`? Use el camino más corto posible.

     ```
     cd ~michael/Music
     ```

2. Considere la siguiente salida de `ls -lh` para responder las siguientes dos preguntas. Tenga en cuenta que los directorios se indican con una `d` al comienzo de la línea.

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-
24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-
07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-
30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

   - Cuando ejecutas el comando `ls -lrS`, ¿Qué archivo estará al principio?

     Las tres carpetas son todas de 4.0K, que es el tamaño de archivo más pequeño. Luego el comando `ls` ordenará por defecto los directorios alfabéticamente. La respuesta correcta es el archivo `scary.jpg`.

   - Describa lo que espera ver al ejecutar `ls -ad */`.

     Este comando mostrará todos los subdirectorios incluidos los ocultos.

Respuestas a los ejercicios exploratorios

1. Ejecute el comando `ls -lh` en un directorio que contenga subdirectorios. Tenga en cuenta el tamaño indicado de estos directorios. ¿Te parecen correctos estos tamaños de archivo? ¿Representan con precisión el contenido de todos los archivos dentro de ese directorio?

   No, cada directorio tiene un tamaño de archivo listado de 4096 bytes. Esto se debe a que los directorios son una abstracción: ellos no existen como una estructura de árbol en el disco. Cuando observas un directorio en la lista, esta observando un *enlace* a una lista de archivos. El tamaño de estos enlaces es de 4096 bytes.

2. Aquí hay un nuevo comando para probar: `du -h`. Ejecute este comando y describa el resultado que le proporciona.

   El comando `du` generará una lista de todos los archivos y directorios e indicará el tamaño de cada uno. Por ejemplo, `du -s` mostrará el tamaño de todos los archivos, directorios y subdirectorios para una determinada ubicación.

3. En muchos sistemas Linux, puedes escribir `ll` y obtener el mismo resultado como si escribieras `ls -l`. Sin embargo, tenga en cuenta que `ll` *no* es un comando. Por ejemplo, `man ll` le dará el mensaje de que no existe un manual para esto. Este es un ejemplo de un *alias*. ¿Por qué los alias podrían ser útiles para un usuario?

   `ll` es un *alias* de `ls -l`. En Bash podemos usar alias para simplificar los comandos de uso común. `ll` a menudo se define por usted en Linux, pero también puede crear sus propios alias.



2.4 Crear, mover y borrar archivos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 2.4

Importancia

2

Áreas de conocimiento clave

- Files and directories
- Case sensitivity
- Simple globbing

Lista parcial de archivos, términos y utilidades

- `mv`, `cp`, `rm`, `touch`
- `mkdir`, `rmdir`



2.4 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	2 Encontrando tu camino en el sistema Linux
Objetivo:	2.4 Creando, moviendo y borrando archivos
Lección:	1 de 1

Introducción

Esta lección cubrirá la administración de archivos y directorios en Linux usando herramientas de línea de comandos.

Un archivo es una colección de datos con un nombre y un conjunto de atributos. Si por ejemplo, usted tuviera que transferir algunas fotos desde su teléfono a una computadora y darles nombres un descriptivo; llegaría a tener varios archivos de imágenes con atributos como la hora la cual accedió por última vez o se modificó.

Un directorio es un tipo especial de archivo utilizado para organizar otros archivos. Una buena manera de pensar en los directorios es como las carpetas que se usan para organizar papeles en un organizador de archivos, a diferencia de las carpetas de archivos de papel, usted puede mover fácilmente directorios dentro de otros directorios.

La línea de comandos es la forma más efectiva de administrar archivos en un sistema Linux, ya que la shell y las herramientas de línea de comandos tienen características que hacen que su uso sea más rápido y fácil que el de un administrador gráfico de archivos.

En esta sección usted usará los comandos `ls`, `mv`, `cp`, `pwd`, `find`, `touch`, `rm`, `rmdir`, `echo`, `cat`, y `mkdir`

para administrar y organizar archivos así como directorios.

Sensibilidad a las mayúsculas y minúsculas

A diferencia de Microsoft Windows, los nombres de archivos y directorios en sistemas Linux se distinguen entre mayúsculas y minúsculas; lo que significa que los nombres `/etc/` y `/ETC/` son directorios diferentes:

```
$ cd /
$ ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
$ cd ETC
bash: cd: ETC: No such file or directory
$ pwd
/
$ cd etc
$ pwd
/etc
```

El `pwd` muestra el directorio en el que se encuentra actualmente, como puede observar, el cambio a `/ETC` no funcionó, ya que dicho directorio no existe, sin embargo el cambio al directorio `/etc` que sí existe tuvo éxito.

Creación de directorios

El comando `mkdir` se usa para crear directorios.

Vamos a crear un nuevo directorio dentro de nuestro directorio home:

```
$ cd ~
$ pwd
/home/user
$ ls
Desktop  Documents  Downloads
$ mkdir linux_essentials-2.4
$ ls
Desktop  Documents  Downloads  linux_essentials-2.4
$ cd linux_essentials-2.4
$ pwd
/home/emma/linux_essentials-2.4
```

Durante de esta lección, todos los comandos tendrán lugar dentro de este directorio o en uno de sus subdirectorios.

Para volver fácilmente al directorio de la lección desde cualquier otra ubicación puede usar el comando:

```
$ cd ~/linux_essentials-2.4
```

El shell interpreta el carácter `~` como su directorio raíz.

Cuando esté en el directorio de la lección, cree más directorios que usaremos para los ejercicios, usted puede crear todos los nombres de los directorios separados por espacios con el comando `mkdir`:

```
$ mkdir creating moving copying/files copying/directories deleting/directories
deleting/files globs
mkdir: cannot create directory ‘copying/files’: No such file or directory
mkdir: cannot create directory ‘copying/directories’: No such file or directory
mkdir: cannot create directory ‘deleting/directories’: No such file or directory
mkdir: cannot create directory ‘deleting/files’: No such file or directory
$ ls
creating  globs  moving
```

Observe el mensaje de error y notará que sólo se crearon `moving`, `globs`, y `creating`. Los directorios `copying` y `deleting` no existen todavía. Por defecto, el comando `mkdir`, no creará un directorio dentro de un directorio que no existe. La opción `-p` o `--parents` ordena a `mkdir` crear directorios padre si no existen. Intente nuevamente usar el comando `mkdir`, pero con la opción `-p`:

```
$ mkdir -p creating moving copying/files copying/directories deleting/directories
deleting/files globs
```

Ahora no recibirás ningún mensaje de error. Veamos qué directorios existen ahora:

```
$ find
.
./creating
./moving
./globs
./copying
./copying/files
./copying/directories
./deleting
./deleting/directories
./deleting/files
```

El comando `find` se utiliza generalmente para buscar archivos y directorios, pero sin opciones. Su resultado mostrará un listado de todos los archivos, directorios y subdirectorios de su directorio actual.

> **TIP** Al listar el contenido de un directorio con `ls`, las opciones `-t` y `-r` son particularmente útiles. Clasifican la salida por tiempo (`-t`) e invierten su orden (`-r`). En este caso los archivos más recientes estarán en la parte inferior de la salida de comandos.

Creación de archivos

Típicamente, los archivos son creados por los programas que trabajan con los datos guardados en ellos, un archivo vacío puede ser creado usando el comando `touch`. Si ejecuta `touch` en un archivo existente, el contenido del archivo no será cambiado, pero la fecha y hora de modificación de los archivos será actualizada.

Ejecute el siguiente comando para crear algunos archivos para la lección de globbing:

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Ahora vamos a verificar que todos los archivos existen en el directorio `globs`:

```
$ cd globs
$ ls
question1   question14    question23  star1100  star2013
question13  question2012  star10      star2002
```

Puedes revisar el contenido de un archivo de texto con el comando `cat`, pruébelo en uno de los archivos que acabas de crear:

```
$ cat question14
```

Dado que `touch` crea archivos vacíos no debería obtener ninguna salida. Puede usar `echo` con `>` para crear archivos de texto simples. Intento esto:

```
$ echo hello > question15
$ cat question15
hello
```

`echo` muestra el texto en la línea de comandos. El carácter `>` instruye al shell a escribir la salida de un comando en el archivo especificado en lugar de su terminal. En este caso, la salida de `echo`, escribe la palabra `hello` en el archivo `question15`. Esto no es específico de `echo`, puede ser usado con cualquier comando.

> Tenga cuidado al usar `>` si el archivo nombrado ya existe, será sobreescrito!

Cambiando el nombre de los archivos

Los archivos se pueden mover o renombran con el comando `mv`.

Establezca su directorio de trabajo en el directorio `moving`:

```
$ cd ~/linux_essentials-2.4/moving
```

Cree algunos archivos con los que pueda practicar, ya a este punto debería estar familiarizado con estos comandos:

```
$ touch file1 file22
$ echo file3 > file3
$ echo file4 > file4
$ ls
file1  file22  file3  file4
```

Supongamos que `file22` es un error tipográfico y debería ser `file2`. Arréglalo con el comando `mv`. Al

renombrar un archivo, el primer argumento es el nombre actual, el segundo es el nuevo nombre:

```
$ mv file22 file2
$ ls
file1  file2  file3  file4
```

Tenga cuidado con el comando `mv`. Si cambia el nombre de un archivo al nombre de un archivo existente, se sobrescribirá:

```
$ cat file3
file3
$ cat file4
file4
$ mv file4 file3
$ cat file3
file4
$ ls
file1  file2  file3
```

Observe como el contenido de `file3` es ahora `file4`. Use la opción `-i` para hacer que `mv` le pregunte si está a punto de sobreescribir un archivo existente:

```
$ touch file4 file5
$ mv -i file4 file3
mv: overwrite ‘file3’? y
```

Moviendo archivos

Los archivos se mueven de un directorio a otro con el comando `mv`.

Cree unos cuantos directorios para mover algunos archivos:

```
$ cd ~/linux_essentials-2.4/moving
$ mkdir dir1 dir2
$ ls
dir1  dir2  file1  file2  file3  file5
```

Mover `file1` a `dir1`:

```
$ mv file1 dir1
$ ls
dir1  dir2  file2  file3  file5
$ ls dir1
file1
```

Observe como el último argumento de `mv` es el directorio destino. Cuando el último argumento de `mv` es un directorio, los archivos son movidos dentro de este; se puede especificar múltiples archivos en un solo comando `mv`:

```
$ mv file2 file3 dir2
$ ls
dir1  dir2  file5
$ ls dir2
file2  file3
```

También es posible usar `mv` para mover y renombrar directorios. Renombre `dir1` a `dir3`:

```
$ ls
dir1  dir2  file5
$ ls dir1
file1
$ mv dir1 dir3
$ ls
dir2  dir3  file5
$ ls dir3
file1
```

Eliminando archivos y directorios

El comando `rm` puede eliminar archivos y directorios. Mientras que el comando `rmdir` sólo puede eliminar directorios:

```
$ cd ~/linux_essentials-2.4/moving
$ ls
dir2  dir3  file5
$ rmdir file5
rmdir: failed to remove ‘file5’: Not a directory
$ rm file5
$ ls
dir2  dir3
```

Por defecto `rmdir` sólo puede borrar directorios vacíos, por lo tanto tuvimos que usar `rm` para borrar un archivo regular:

```
$ cd ~/linux_essentials-2.4/
$ ls
copying  creating  deleting  globs  moving
$ rmdir deleting
rmdir: failed to remove ‘deleting’: Directory not empty
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 files
```

Por defecto, `rmdir` no elimina un directorio que no esté vacío, utilice `rmdir` para eliminar uno de los subdirectorios vacíos del directorio `deleting`:

```
$ ls -a deleting/files
.  ..
$ rmdir deleting/files
$ ls -l deleting
directories
```

Eliminar grandes cantidades de archivos o estructuras de directorios con muchos subdirectorios puede parecer tedioso, pero en realidad es fácil. Por defecto, el comando `rm` sólo funciona con archivos normales, la opción `-r` se utiliza para anular este comportamiento, pero ¡Cuidado, `rm -r` podría darte un mal día! Cuando uses la opción `-r`, `rm` no sólo borrará todos los directorios, sino también todo lo que haya dentro de ese directorio, incluyendo los subdirectorios y sus contenidos:

```
$ ls
copying  creating  deleting  globs  moving
$ rm deleting
rm: cannot remove ‘deleting’: Is a directory
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
$ rm -r deleting
$ ls
copying  creating  globs  moving
```

Observe como `deleting` ha desaparecido, a pesar de que no estaba vacío? Al igual que `mv`, `rm` tiene una opción `-i` para avisar antes de hacer algo. Use `rm -ri` para eliminar directorios de la sección `moving` que ya no son necesarios:

```
$ find
.
./creating
./moving
./moving/dir2
./moving/dir2/file2
./moving/dir2/file3
./moving/dir3
./moving/dir3/file1
./globs
./globs/question1
./globs/question2012
./globs/question23
./globs/question13
./globs/question14
./globs/star10
./globs/star1100
./globs/star2002
./globs/star2013
./globs/question15
./copying
./copying/files
./copying/directories
$ rm -ri moving
rm: descend into directory ‘moving’? y
rm: descend into directory ‘moving/dir2’? y
rm: remove regular empty file ‘moving/dir2/file2’? y
rm: remove regular empty file ‘moving/dir2/file3’? y
rm: remove directory ‘moving/dir2’? y
rm: descend into directory ‘moving/dir3’? y
rm: remove regular empty file ‘moving/dir3/file1’? y
rm: remove directory ‘moving/dir3’? y
rm: remove directory ‘moving’? y
```

Copiando archivos y directorios

El comando `cp` se utiliza para copiar archivos y directorios. Copie algunos archivos en el directorio `copying`:


```
$ cd ~/linux_essentials-2.4/copying
$ ls
directories  files
$ cp /etc/nsswitch.conf files/nsswitch.conf
$ cp /etc/issue /etc/hostname files
```

Si el último argumento es un directorio, `cp` creará una copia de los argumentos anteriores dentro de ese directorio. Al igual que `mv`, se puede especificar múltiples archivos a la vez siempre y cuando el objetivo sea un directorio.

Cuando ambos operandos de `cp` son archivos y existen ambos archivos, `cp` sobrescribe el segundo archivo con una copia del primero:

```
$ cd ~/linux_essentials-2.4/copying/files
$ ls
hostname  issue  nsswitch.conf
$ cat hostname
mycomputer
$ cat issue
Debian GNU/Linux 9 \n \l

$ cp hostname issue
$ cat issue
mycomputer
```

Ahora vamos a intentar crear una copia del directorio `files` dentro del directorio `directories`:

```
$ cd ~/linux_essentials-2.4/copying
$ cp files directories
cp: omitting directory ‘files’
```

Como puede observar, `cp` por defecto sólo funciona en archivos individuales, para copiar un directorio utilice la opción `-r`. Tenga en cuenta que la opción `-r` hará que `cp` también copie el contenido del directorio que está copiando:

```
$ cp -r files directories
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
```

Observe cómo un directorio existente fue usado como destino. El comando `cp` crea una copia del directorio fuente dentro de él? Si el destino no existe, lo creará con el contenido del directorio fuente:

```
$ cp -r files files2
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
./files2
./files2/nsswitch.conf
./files2/fstab
./files2/hostname
```

Globbing

Lo que comúnmente se conoce como globbing hace referencia a un lenguaje simple de coincidencia de patrones. Los shells utilizan este lenguaje para referirse a grupos de archivos cuyos nombres coinciden con un patrón específico. POSIX.1-2017 especifica los siguientes caracteres que coinciden con un patrón:

`*`
: Coincide con cualquier número de caracteres, incluyendo los no caracteres.

`?`
: Coincide con cualquier carácter.

`[]`
: Corresponde a una clase de caracteres.

En español, esto significa que puede decirle a su shell que coincida con un patrón en lugar de una cadena literal de texto. Los usuarios de Linux suelen especificar varios archivos con un “glob” en lugar de escribir el nombre de cada archivo:

```
$ cd ~/linux_essentials-2.4/globs
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls star1*
star10  star1100
$ ls star*
star10  star1100  star2002  star2013
$ ls star2*
star2002  star2013
$ ls star2*2
star2002
$ ls star2013*
star2013
```

Shell expande `*` a cualquier número de caracteres, por tanto interpreta que `star*` significa cualquier nombre (archivo o directorio) que coincidan(al inicio) con esos caracteres. Cuando ejecutas el comando `ls star*`, shell no ejecuta el programa `ls` con un argumento de `star*`, sino que busca archivos en el directorio actual que concuerden con el patrón `star` (incluyendo sólo `star`). Además convierte cada archivo que coincida con el patrón en un argumento de `ls`:

```
$ ls star*
```

En lo que al comando `ls` se refiere, esto es equivalente a:

```
$ ls star10  star1100  star2002  star2013
```

El carácter `*` no significa nada para `ls`. Para probar esto ejecute el siguiente comando:

```
$ ls star\*
ls: cannot access star*: No such file or directory
```

Cuando se precede un carácter con un `\`, estás instruyendo a su shell para que no lo interprete. En este caso, usted desea que `ls` tenga un argumento de `star*`.

El carácter `?` se expande a cualquier personaje, pruebe los siguientes comandos:

```
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls question?
question1
$ ls question1?
question13  question14  question15
$ ls question?3
question13  question23
$ ls question13?
ls: cannot access question13?: No such file or directory
```

Los corchetes `[]` se usan para igualar rangos o clases de caracteres. Estos funcionan como lo hacen en las expresiones regulares POSIX, excepto que con globs se usa `^` en lugar de `!`.

Cree algunos archivos para experimentar:

```
$ mkdir brackets
$ cd brackets
$ touch file1 file2 file3 file4 filea fileb filec file5 file6 file7
```

Los rangos entre `[]` corchetes se expresan mediante un `-`:

```
$ ls
file1  file2  file3  file4  file5  file6  file7  filea  fileb  filec
$ ls file[1-2]
file1  file2
$ ls file[1-3]
file1  file2  file3
```

Se pueden especificar varios rangos:

```
$ ls file[1-25-7]
file1  file2  file5  file6  file7
$ ls file[1-35-6a-c]
file1  file2  file3  file5  file6  filea  fileb  filec
```

Los corchetes también se pueden utilizar para hacer coincidir un conjunto específico de caracteres.

```
$ ls file[1a5]
file1  file5  filea
```

También puede utilizar el carácter `^` como primer carácter para que coincida con todo, excepto con ciertos caracteres.

```
$ ls file[^a]
file1  file2  file3  file4  file5  file6  file7  fileb  filec
```

Lo último que cubriremos en esta lección son las clases de caracteres. Para que coincidan con una clase de caracteres, use `[:classname:]`. Por ejemplo, para usar la clase de dígitos que coincida con los números, debes hacer algo como esto:

```
$ ls file[[:digit:]]
file1  file2  file3  file4  file5  file6  file7
$ touch file1a file11
$ ls file[[:digit:]a]
file1  file2  file3  file4  file5  file6  file7  filea
$ ls file[[:digit:]]a
file1a
```

El glob `file[[:digit:]a]`, coincide con `file` seguido de un dígito o `a`.

POSIX requiere las siguientes clases de caracteres para todas las localizaciones:

`[:alnum:]`
: Letras y números.

`[:alpha:]`
Letras mayúsculas o minúsculas.

`[:blank:]`
Espacios y tabulaciones.

`[:cntrl:]`
Caracteres de control, por ejemplo, backspace, bell, NAK, escape.

`[:digit:]`
Números (`012345456789`).

`[:graph:]`
Caracteres gráficos (todos los caracteres excepto `ctrl` y el carácter de espacio)

`[:lower:]`
Letras minúsculas (`a-z`).

`[:print:]`
Caracteres imprimibles (`alnum`, `punct` y el carácter del espacio).

`[:punct:]`
Caracteres de puntuación, es decir `!`, `&`, `"`.

`[:space:]`
Caracteres de espacio en blanco, por ejemplo, tabulaciones, espacios, líneas nuevas.

`[:upper:]`
Letras mayúsculas (`A-Z`).

`[:xdigit:]`
Números hexadecimales (normalmente `0123456789abcdefABCDEF`).

Ejercicios guiados

1. Dado lo siguiente, seleccione los directorios que podría crear el comando `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

`/tmp`	
`/tmp/outfiles`	
`/tmp/outfiles/text`	
`/tmp/outfiles/text/today`	
`/tmp/infiles`	
`/tmp/infiles/text`	
`/tmp/infiles/text/today`	

2. ¿Qué hace `-v` para `mkdir`, `rm` y `cp`?

3. ¿Qué sucede si accidentalmente intenta copiar tres archivos en la misma línea de comando a un archivo que ya existe en lugar de un directorio?

4. ¿Qué sucede cuando usa `mv` para mover un directorio dentro de sí mismo?

5. ¿Cómo eliminaría todos los archivos en su directorio actual que comienzan con `old`?

6. ¿Cuál de los siguientes archivos `log_[a-z]_201?_*_01.txt` coinciden?

`log_3_2017_Jan_01.txt`	
`log_+_2017_Feb_01.txt`	
`log_b_2007_Mar_01.txt`	

`log_f_201A_Wednesday_01.txt`	

7. Cree algunos globs para que coincidan con la siguiente lista de nombres de archivo:

```
doc100
doc200
doc301
doc401
```


Ejercicios exploratorios

1. Utilice la página del manual `cp` para descubrir cómo hacer una copia de un archivo y hacer que los permisos y la hora de modificación coincidan con el original.

2. ¿Qué hace el comando `rmdir -p`? Pruébelo y explique cómo difiere de `rm -r`.

3. NO EJECUTE REALMENTE ESTE COMANDO: ¿Qué cree que hará `rm -ri /*` ? (Honestamente, ¡no intentes hacer esto!)

4. Además de usar `-i`. ¿Es posible evitar que `mv` sobrescriba los archivos de destino?

5. Explique el comando `cp -u`.

Resumen

El entorno de línea de comandos de Linux proporciona herramientas para administrar archivos. Algunos de los más utilizados son `cp`, `mv`, `mkdir`, `rm` y `rmdir`. Estas herramientas combinadas con globs, permiten a los usuarios mejorar su trabajo.

Muchos comandos tienen una opción `-i` que pregunta antes de hacer algo. Esto podrían ahorrarte muchos problemas si escribes algo mal.

Muchos comandos tienen una opción `-r`. La opción `-r` generalmente significa recursión. En matemáticas e informática, una función recursiva es una función que se utiliza a sí misma en su definición. Cuando se trata de herramientas de línea de comandos, generalmente significa aplicar el comando a un directorio y todo lo que contiene.

Comandos usados en esta lección:

`cat`
: Muestra o concatena el contenido de ficheros.

`cp`
: Copia archivos o directorios.

`echo`
: Salida de texto.

`find`
: Recorre un árbol del sistema de archivos y busca archivos que coincidan con un conjunto específico de criterios.

`ls`
: Muestra propiedades de archivos y directorios, y enumera los contenidos de un directorio.

`mkdir`
: Crea nuevos directorios.

`mv`
: Mueve o renombra archivos o directorios.

`pwd`
: Salida del directorio de trabajo actual.

`rm`
: Eliminar archivos o directorios.

`rmdir`
: Elimina directorios.

`touch`
: Crea nuevos archivos vacíos o actualiza el tiempo de modificación de un archivo existente.

Respuestas a los ejercicios guiados

1. Dado lo siguiente, seleccione los directorios que podría crear el comando `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

   ```
   $ pwd
   /tmp
   $ find
   .
   ./outfiles
   ./outfiles/text
   ```

   Se crearían los directorios marcados. Los directorios `/tmp`, `/tmp/outfiles` y `/tmp/outfiles/text` ya existen, por lo que `mkdir` los ignorará.

`/tmp`	
`/tmp/outfiles`	
`/tmp/outfiles/text`	
`/tmp/outfiles/text/today`	X
`/tmp/infiles`	X
`/tmp/infiles/text`	X
`/tmp/infiles/text/today`	X

2. ¿Qué hace `-v` para `mkdir`, `rm` y `cp`?

   Por lo general, `-v` activa la salida detallada. Hace que los respectivos programas muestren lo que están haciendo a la hora de ser ejecutados.

   ```
   $ rm -v a b
   removed 'a'
   removed 'b'
   $ mv -v a b
   'a' -> 'b'
   $ cp -v b c
   'b' -> 'c'
   ```

3. ¿Qué sucede si accidentalmente intenta copiar tres archivos en la misma línea de comando a un archivo que ya existe en lugar de un directorio?

`cp` se negará a hacer la operación y generará un mensaje de error:

```
$ touch a b c d
$ cp a b c d
cp: target 'd' is not a directory
```

4. ¿Qué sucede cuando usa `mv` para mover un directorio dentro de sí mismo?

   Recibirá un mensaje de error que indica que `mv` no podrá hacer eso.

```
$ mv a a
mv: cannot move 'a' to a subdirectory of itself, 'a/a'
```

5. ¿Cómo eliminaría todos los archivos en su directorio actual que comienzan con `old`?

   Usaría el glob `old *` con `rm`:

```
$ rm old*
```

6. ¿Cuál de los siguientes archivos `log_[a-z]_201?_*_01.txt` coinciden?

`log_3_2017_Jan_01.txt`	
`log_+_2017_Feb_01.txt`	
`log_b_2007_Mar_01.txt`	
`log_f_201A_Wednesday_01.txt`	X

```
$ ls log_[a-z]_201?_*_01.txt
log_f_201A_Wednesday_01.txt
```

`log_ [a-z]` coincide con `log_` seguido de cualquier letra minúscula, por lo que tanto `log_f_201A_Wednesday_01.txt` como `log_b_2007_Mar_01.txt` coinciden. `_201?` coincide con cualquier carácter individual, por lo que solo `log_f_201A_Wednesday_01.txt` coincide. Finalmente `* _01.txt` coincide con todo lo que termina con `_01.txt`, por lo que nuestra opción restante coincide.

7. Create a few globs to match the following list of file names:



```
doc100
doc200
doc301
doc401
```

There are several solutions. Here are some of them:

```
doc*
doc[1-4]*
doc?0?
doc[1-4]0?
```

Respuestas a los ejercicios exploratorios

1. Utilice la página del manual `cp` para descubrir cómo hacer una copia de un archivo y hacer que los permisos y la hora de modificación coincidan con el original.

   Usaría la opción `-p`. Desde la página del manual dice lo siguiente:

   ```
   $ man cp
   -p     same as --preserve=mode,ownership,timestamps
   --preserve[=ATTR_LIST]
                 preserve the specified attributes (default: mode,ownership,time-
                 stamps), if  possible  additional  attributes:  context,  links,
                 xattr, all
   ```

2. ¿Qué hace el comando `rmdir -p`? Pruébelo y explique cómo difiere de `rm -r`.

   Hace que `rmdir` se comporte de manera similar a `mkdir -p`. Si pasa un árbol de directorios vacíos, los eliminará a todos.

   ```
   $ find
   .
   ./a
   ./a/b
   ./a/b/c
   $ rmdir -p a/b/c
   $ ls
   ```

3. NO EJECUTE REALMENTE ESTE COMANDO: ¿Qué cree que hará `rm -ri /*` ? (Honestamente, ¡no intentes hacer esto!)

   Eliminará todos los archivos y directorios que pueda escribir su cuenta de usuario. Esto incluye cualquier sistema de archivos de red.

4. Además de usar `-i`. ¿Es posible evitar que `mv` sobrescriba los archivos de destino?

   Sí, la opción `-n` o `--no-clobber` evita que `mv` sobrescriba los archivos.

```
$ cat a
a
$ cat b
b
$ mv -n a b
$ cat b
b
```

5. Explique `cp -u`.

   La opción `-u` hace que `cp` sólo copie un archivo si el destino no existe o es más antiguo que el archivo de origen.

```
$ ls -l
total 24K
drwxr-xr-x 123 emma student  12K Feb  2 05:34 ..
drwxr-xr-x   2 emma student 4.0K Feb  2 06:56 .
-rw-r--r--   1 emma student    2 Feb  2 06:56 a
-rw-r--r--   1 emma student    2 Feb  2 07:00 b
$ cat a
a
$ cat b
b
$ cp -u a b
$ cat b
b
$ cp -u a c
$ ls -l
total 12
-rw-r--r-- 1 emma student 2 Feb  2 06:56 a
-rw-r--r-- 1 emma student 2 Feb  2 07:00 b
-rw-r--r-- 1 emma student 2 Feb  2 07:00 c
```




Tema 3: El poder de la línea de comandos



3.1 Archivar ficheros desde la línea de comandos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 3.1

Importancia

2

Áreas de conocimiento clave

- Ficheros, directorios
- Archivadores, compresión

Lista parcial de archivos, términos y utilidades

- `tar`
- Opciones comunes del comando `tar`
- `gzip`, `bzip2`, `xz`
- `zip`, `unzip`



3.1 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	3 El poder de la línea de comandos
Objetivo:	3.1 Archivar ficheros desde la línea de comandos
Lección:	1 de 1

Introducción

La compresión consiste en reducir la cantidad de espacio que consume un conjunto de datos específicos. Generalmente se utiliza para disminuir el espacio que se necesita para almacenar un archivo y para reducir la cantidad de datos que se envían a través de una conexión de red.

La compresión funciona reemplazando patrones repetitivos en los datos. Supongamos que tienes una novela. Algunas palabras son extremadamente comunes pero tienen múltiples caracteres, como la palabra “el”. Podría reducir el tamaño de la novela significativamente si reemplazara estas palabras y sus patrones comunes en un solo carácter. Por ejemplo, reemplace “el” con una letra griega que no se usa en ninguna otra parte del texto. Tomando el ejemplo anterior, se puede decir que los algoritmos de compresión tratan los datos de manera similar, pero de una forma más compleja.

La compresión tiene dos variantes, *sin pérdida (lossless)* y *con pérdida (lossy)*. La información comprimida con un algoritmo "lossless", puede ser descomprimida en su forma original. Los datos comprimidos con un algoritmo "lossy" no pueden ser recuperados. Los algoritmos "lossy" se utilizan generalmente en imágenes, video y audio donde la pérdida de calidad es imperceptible para los seres humanos, irrelevante para el contexto o cuando se aprovecha una mejora en el rendimiento de la red.

Las herramientas de archivo (Archiving tools) se utilizan para agrupar archivos y directorios en un

solo fichero. Algunos usos comunes son las copias de seguridad, la agrupación del código fuente y la retención de datos.

El archivo y la compresión se usan comúnmente juntos. Incluso algunas herramientas de archivo comprimen su contenido de forma predeterminada; otros pueden comprimir opcionalmente sus contenidos. Si desea comprimir el contenido se deben usar algunas herramientas de archivo junto con herramientas de compresión independientes.

La herramienta `tar` es la más común para archivar información en Linux. La mayoría de las distribuciones de Linux incluyen la versión GNU de `tar`, por lo que se tratará en esta lección. Por sí solo este programa archiva información sin usar compresión.

Existen muchas herramientas de compresión en Linux, algunas de las más comunes son `bzip2`, `gzip`, y `xz`. Es posible encontrar las tres en la mayoría de los sistemas, aunque pueden existir sistemas antiguos o muy mínimos donde `xz` o `bzip` no están instalados. Los usuarios regulares de Linux manejan archivos comprimidos con cualquiera de las tres herramientas, las tres usan algoritmos diferentes, por lo que un archivo comprimido con una herramienta no puede ser descomprimido por otra. La compresión trae consigo algunas desventajas; si se desea una tasa de compresión alta, tomará más tiempo para comprimir y descomprimir el archivo, ya que se requiere más procesamiento para encontrar patrones más complejos. Todas estas herramientas comprimen información pero no pueden crear archivos que contengan múltiples ficheros.

Las herramientas independientes de compresión generalmente no están disponibles en sistemas Windows. Las herramientas de compresión y archivado de Windows suelen estar juntas. Es importante tener esto en cuenta si se tienen sistemas Linux y Windows que necesiten compartir archivos.

Las herramientas `zip` y `unzip` están disponibles en sistemas Linux para manejar archivos `.zip` que comúnmente son usados en sistemas Windows. Estas herramientas no están instaladas por defecto en todos los sistemas, por lo que si se desea usarlas, necesitará realizar la instalación. Afortunadamente, se encuentran (típicamente) en los repositorios de las distribuciones.

Herramientas de compresión

La cantidad de espacio en disco que se ahorra al comprimir los archivos depende de varios factores: la naturaleza de los datos que se comprimen, el algoritmo utilizado para comprimir los datos y el nivel de compresión. No todos los algoritmos admiten diferentes niveles de compresión.

A continuación se configuran algunos archivos de prueba para comprimir:

```
$ mkdir ~/linux_essentials-3.1
$ cd ~/linux_essentials-3.1
$ mkdir compression archiving
$ cd compression
$ cat /etc/* > bigfile 2> /dev/null
```

Se crean tres copias de este archivo:

```
$ cp bigfile bigfile2
$ cp bigfile bigfile3
$ cp bigfile bigfile4
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile4
```

Ahora se comprimen los archivos con cada una de las herramientas de compresión mencionadas:

```
$ bzip2 bigfile2
$ gzip bigfile3
$ xz bigfile4
$ ls -lh
total 1.2M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 170K Jun 23 08:08 bigfile2.bz2
-rw-r--r-- 1 emma emma 179K Jun 23 08:08 bigfile3.gz
-rw-r--r-- 1 emma emma 144K Jun 23 08:08 bigfile4.xz
```

Compare los tamaños de los archivos comprimidos con el archivo sin comprimir denominado `bigfile`. Observe también cómo las herramientas de compresión agregaron extensiones a los nombres de archivo y eliminaron los archivos sin comprimir.

Para descomprimir los archivos se utilizan `bunzip2`, `gunzip`, o `unxz`:

```
$ bunzip2 bigfile2.bz2
$ gunzip bigfile3.gz
$ unxz bigfile4.xz
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile4
```

Nuevamente se observa que el archivo comprimido se borra una vez que se descomprime.

Algunas herramientas soportan diferentes niveles de compresión, normalmente un nivel más alto requiere más memoria y ciclos de CPU, pero se obtiene un archivo comprimido más pequeño; lo contrario es cierto para un nivel más bajo. A continuación se muestra un ejemplo utilizando `xz` y `gzip`:

```
$ cp bigfile bigfile-gz1
$ cp bigfile bigfile-gz9
$ gzip -1 bigfile-gz1
$ gzip -9 bigfile-gz9
$ cp bigfile bigfile-xz1
$ cp bigfile bigfile-xz9
$ xz -1 bigfile bigfile-xz1
$ xz -9 bigfile bigfile-xz9
$ ls -lh bigfile bigfile-* *
total 3.5M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 205K Jun 23 13:14 bigfile-gz1.gz
-rw-r--r-- 1 emma emma 178K Jun 23 13:14 bigfile-gz9.gz
-rw-r--r-- 1 emma emma 156K Jun 23 08:08 bigfile-xz1.xz
-rw-r--r-- 1 emma emma 143K Jun 23 08:08 bigfile-xz9.xz
```

No es necesario descomprimir un archivo cada vez que se necesite. Las herramientas de compresión, normalmente incluyen versiones especiales de aplicativos que son usados para leer archivos de texto, por ejemplo, `gzip` incluye una versión de `cat`, `grep`, `diff`, `less`, `more`, etc. Para `gzip`, las herramientas utilizan el prefijo `z`, mientras que el prefijo `bz` se usa para `bzip2` y `xz` para `xz`. A continuación se muestra un ejemplo del uso de `zcat` para mostrar el contenido de un archivo comprimido con `gzip`:


```
$ cp /etc/hosts ./
$ gzip hosts
$ zcat hosts.gz
127.0.0.1   localhost

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Archivadores

El programa `tar` es probablemente el archivador más utilizado en sistemas Linux. Su nombre proviene de la abreviatura de “tape archive”, ya que los archivos creados con `tar` se denominan a menudo como *tar balls*. Es muy común que el código fuente de las aplicaciones se distribuya en *tar balls*.

Las distribuciones de Linux que incluyen la versión GNU de `tar` tiene muchas opciones, esta lección cubrirá el subconjunto más utilizado.

A continuación se crea un archivo de los ficheros usados para la compresión:

```
$ cd ~/linux_essentials-3.1
$ tar cf archiving/3.1.tar compression
```

La opción `c` indica a `tar` que cree un nuevo archivo y la opción `f` el nombre del archivo a crear. El argumento que sigue después de las opciones siempre será el nombre del archivo con el que se va a trabajar. El resto de los argumentos son las rutas a cualquier fichero o directorio que se desee añadir, listar o extraer del archivo. En el ejemplo, se añade el directorio `compression` y todo su contenido al archivo comprimido.

Para ver el contenido de un archivo creado con `tar`, se utiliza la opción `t`:

```
$ tar -tf 3.1.tar
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
```

Nótese cómo las opciones van precedidas de `-`. A diferencia de la mayoría de los programas, con `tar`, el `-` no es necesario cuando se especifican opciones, aunque no causa ningún problema si se usa.

NOTE Se puede usar la opción `-v` para permitir que `tar` muestre los nombres de los archivos en los que opera cuando crea o extrae un archivo.

A continuación se extrae el archivo:

```
$ cd ~/linux_essentials-3.1/archiving
$ ls
3.1.tar
$ tar xf 3.1.tar
$ ls
3.1.tar  compression
```

Si se necesita solamente un fichero fuera del archivo comprimido, se puede especificar después del nombre del archivo y, si es necesario, se pueden especificar varios ficheros:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ ls
3.1.tar
$ tar xvf 3.1.tar compression/hosts.gz
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
$ ls
3.1.tar  compression
$ ls compression
hosts.gz
```

Con la excepción de las rutas absolutas (rutas que empiezan por `/`), los archivos `tar` conservan la ruta completa de los ficheros cuando son creados. Ya que el archivo `3.1.tar` fue creado con un solo directorio, ese directorio será creado en relación a su directorio de trabajo actual cuando sea extraído. El siguiente ejemplo describe lo anterior:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ cd ../compression
$ tar cf ../tar/3.1-nodir.tar *
$ cd ../archiving
$ mkdir untar
$ cd untar
$ tar -xf ../3.1-nodir.tar
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
```

TIP Si se desea utilizar la ruta absoluta en un archivo `tar`, se debe usar la opción `P`. Es importante tener en cuenta que esta operación puede sobrescribir ficheros importantes y puede causar errores en el sistema.

El programa `tar` también puede gestionar la compresión y descompresión de archivos sobre la marcha. `tar` lo hace llamando a una de las herramientas de compresión presentadas anteriormente. Es tan simple como añadir la opción apropiada para el algoritmo de compresión. Las más utilizadas son `j`, `J`, y `z` para `bzip2`, `xz`, y `gzip`, respectivamente. A continuación se muestran ejemplos utlizando los algoritmos mencionados anteriormente:

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
$ tar -czf gzip.tar.gz bigfile bigfile2 bigfile3
$ tar -cjf bzip2.tar.bz2 bigfile bigfile2 bigfile3
$ tar -cJf xz.tar.xz bigfile bigfile2 bigfile3
$ ls -l | grep tar
-rw-r--r-- 1 emma emma  450202 Jun 27 05:56 bzip2.tar.bz2
-rw-r--r-- 1 emma emma  548656 Jun 27 05:55 gzip.tar.gz
-rw-r--r-- 1 emma emma  147068 Jun 27 05:56 xz.tar.xz
```

Nótese que los archivos `.tar` tienen tamaños diferentes, esto demuestra que fueron comprimidos satisfactoriamente. Si se crean archivos `.tar` comprimidos, siempre se debe añadir una segunda extensión de archivo que indique el algoritmo utilizado: `.xz`, `.bz`, y `.gz` para `xz`, `bzip2`, y `gzip`, respectivamente. También es posible utilizar extensiones más cortas como `.tgz`.

Es posible añadir ficheros a archivos tar sin comprimir. Para esto se utiliza la opción `u`. Si se intenta añadir a un archivo comprimido, se obtendrá un error.


```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile   bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  bzip2.tar.bz2  hosts.gz
bigfile2  bigfile4  bigfile-gz9.gz  bigfile-xz9.xz  gzip.tar.gz    xz.tar.xz
$ tar cf plain.tar bigfile bigfile2 bigfile3
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
$ tar uf plain.tar bigfile4
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
bigfile4
$ tar uzf gzip.tar.gz bigfile4
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
```

Gestión de archivos ZIP

Los sistemas Windows generalmente no tienen aplicaciones para manejar archivos `.tar`, o muchas de las herramientas de compresión que se encuentran comúnmente en los sistemas Linux. Si se necesita interactuar con sistemas Windows, se pueden utilizar archivos ZIP. Los archivos ZIP son similares a los `tar` comprimidos.

Los programas `zip` y `unzip` pueden ser usados para trabajar con archivos ZIP en sistemas Linux. El ejemplo a continuación muestra lo necesario para comenzar a utilizarlos. Primero se crean un conjunto de ficheros:

```
$ cd ~/linux_essentials-3.1
$ mkdir zip
$ cd zip/
$ mkdir dir
$ touch dir/file1 dir/file2
```

A continuación se utiliza `zip` para empaquetar estos ficheros en un archivo ZIP:

```
$ zip -r zipfile.zip dir
  adding: dir/ (stored 0%)
  adding: dir/file1 (stored 0%)
  adding: dir/file2 (stored 0%)
$ rm -rf dir
```

Finalmente, se descomprime el archivo ZIP:

```
$ ls
zipfile.zip
$ unzip zipfile.zip
Archive:  zipfile.zip
   creating: dir/
 extracting: dir/file1
 extracting: dir/file2
$ find
.
./zipfile.zip
./dir
./dir/file1
./dir/file2
```

Cuando se agregan directorios a archivos ZIP, la opción `-r` permite que `zip` incluya el contenido de un directorio. Si no se especifica esa opción se incluyen directorios vacíos en el archivo ZIP.

Ejercicios guiados

1. ¿Cuáles de las siguientes herramientas se usaron para crear estos archivos?

Nombre de archivo	`tar`	`gzip`	`bzip2`	`xz`
`archive.tar`				
`archive.tgz`				
`archive.tar.xz`				

2. ¿Cuáles de estos ficheros son archivos y cuáles son comprimidos?

Nombre de archivo	Archivado	Comprimido
`file.tar`		
`file.tar.bz2`		
`file.zip`		
`file.xz`		

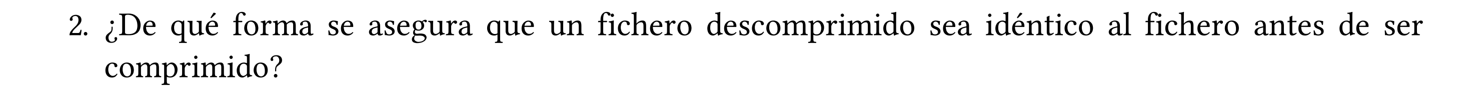
3. ¿Cómo se añadiría un fichero a un archivo `tar` comprimido con `gzip`?

4. ¿Qué opción de `tar` permite incluir el `/` inicial en rutas absolutas?

5. ¿La herramienta `zip` soporta diferentes niveles de compresión?

Ejercicios exploratorios

1. ¿Es posible utilizar comodines utilizando `tar` al extraer archivos?



2. ¿De qué forma se asegura que un fichero descomprimido sea idéntico al fichero antes de ser comprimido?

3. ¿Qué sucede si se intenta extraer un fichero de un archivo `tar` que ya existe en el sistema de ficheros?

4. ¿Cómo es posible extraer el archivo `archive.tgz` sin utiliza la opción `z` de `tar`?

Resumen

Los sistemas Linux incluyen varias herramientas de compresión y archivado, esta lección cubre las más comunes. La herramienta de archivado más común es `tar`. Si es necesario interactuar con sistemas Windows, `zip` y `unzip` pueden crear y extraer archivos ZIP.

El comando `tar` tiene algunas opciones que vale la pena memorizar: `x` para extraer, `c` para crear, `t` para ver el contenido, y `u` para agregar o reemplazar archivos. La opción `v` muestra los archivos que son procesados por `tar` mientras se crea o extrae un archivo.

Generalmente los repositorios de distribuciones de Linux incluyen muchas herramientas de compresión, las más comunes son `gzip`, `bzip2`, y `xz`. Los algoritmos de compresión generalmente soportan diferentes niveles que permiten optimizar el proceso según la velocidad o el tamaño del archivo. Los archivos pueden descomprimirse utilizando `gunzip`, `bunzip2`, y `unxz`.

Las herramientas de compresión generalmente incluyen programas que se comportan como herramientas comunes de archivos de texto, con la diferencia de que funcionan con archivos comprimidos, algunos ejemplos de estas son `zcat`, `bzcat` y `xzcat`. Las herramientas de compresión suelen incluir programas con la funcionalidad de `grep`, `more`, `less`, `diff`, y `cmp`.

Comandos utilizados en los ejercicios:

`bunzip2`
: Descomprime un archivo comprimido con `bzip2`.

`bzcat`
: Muestra el contenido de un archivo comprimido con `bzip`.

`bzip2`
: Comprime archivos usando el algoritmo y formato `bzip2`.

`gunzip`
: Descomprime un archivo comprimido con `gzip`.

`gzip`
: Comprime archivos usando el algoritmo y formato `gzip`.

`tar`
: Crea, actualiza, lista y extrae archivos `tar`.

`unxz`

Descomprime archivos comprimidos con `xz`.

`unzip`

Descomprime y extrae el contenido de un archivo ZIP.

`xz`

Comprime archivos usando el algoritmo y formato `xz`.

`zcat`

Muestra el contenido de un archivo comprimido con `gzip`.

`zip`

Crea y comprime archivos ZIP.

Respuestas a los ejercicios guiados

1. ¿Cuáles de las siguientes herramientas se usaron para crear estos archivos?

Nombre de archivo	`tar`	`gzip`	`bzip2`	`xz`
`archive.tar`	X			
`archive.tgz`	X	X		
`archive.tar.xz`	X			X

2. ¿Cuáles de estos ficheros son archivos y cuáles son comprimidos?

Nombre de archivo	Archivado	Comprimido
`file.tar`	X	
`file.tar.bz2`	X	X
`file.zip`	X	X
`file.xz`		X

3. ¿Cómo se añadiría un fichero a un archivo `tar` comprimido con `gzip`?

   Se descompime el archivo con `gunzip`, y se añade el fichero con `tar uf`, y por último se comprime con `gzip`.

4. ¿Qué opción de `tar` permite incluir el `/` inicial en rutas absolutas?

   La opción `-P`. De la página de `man`:

```
-P, --absolute-names
        Don't strip leading slashes from file names when creating  archives
```

5. ¿La herramienta `zip` soporta diferentes niveles de compresión?

   Sí. Utilizando la opción `-#`, se sustituye `#` con un número del 0 al 9. De la página de `man`:

```
-#
(-0, -1, -2, -3, -4, -5, -6, -7, -8, -9)
        Regule la velocidad de compresión usando el dígito especificado #,
        donde -0 indica que no hay compresión (almacena todos los archivos), -1
indi-
        proporciona la velocidad de compresión más rápida (menos compresión) y
-9
        indica la velocidad de compresión más lenta (compresión óptima,
        ignora la lista de sufijos). El nivel de compresión predeterminado es
-6.

               Aunque todavía se está trabajando, la intención es que esta
configuración
        controle la velocidad de compresión para todos los métodos de
compresión.
        Actualmente solo se controla la deflación.
```

Respuestas a los ejercicios exploratorios

1. ¿Es posible utilizar comodines utilizando `tar` al extraer archivos?

   Sí, utilizando la opción `--wildcards`. `--wildcards` debe especificarse a continuación del archivo `tar` cuando no se utilice `-` en las opciones. Por ejemplo:

   ```
   $ tar xf tarfile.tar --wildcards dir/file*
   $ tar --wildcards -xf tarfile.tar dir/file*
   ```

2. ¿De qué forma se asegura que un fichero descomprimido sea idéntico al fichero antes de ser comprimido?

   No se necesita nada adicional al utilizar las herramientas presentadas en esta lección. Las tres incluyen *checksums* en el formato de archivo los cuales son verificados en el proceso de descompresión.

3. ¿Qué sucede si se intenta extraer un fichero de un archivo `tar` que ya existe en el sistema de ficheros?

   El fichero se sobrescribe con la versión incluída en el archivo `tar`.

4. ¿Cómo es posible extraer el archivo `archive.tgz` sin utiliza la opción `z` de `tar`?

   Descomprimiéndolo primeramente con `gunzip`.

   ```
   $ gunzip archive.tgz
   $ tar xf archive.tar
   ```



3.2 Buscar y extraer datos de los ficheros

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 3.2

Importancia

3

Áreas de conocimiento clave

- Tuberías en la línea de comandos
- Re-dirección de Entrada/Salida
- Expresiones Regulares básicas usando `.`, `[ ]`, `*`, y `?`

Lista parcial de archivos, términos y utilidades

- `grep`
- `less`
- `cat`, `head`, `tail`
- `sort`
- `cut`
- `wc`



3.2 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	3 El poder de la línea de comandos
Objetivo:	3.2 Buscar y extraer datos de los ficheros
Lección:	1 de 2

Introducción

Este laboratorio se enfoca en la redirección o la transmisión de información de una fuente a otra con la ayuda de herramientas específicas. La línea de comandos de Linux redirige la información a través de canales estándares. La entrada estándar (*stdin* o channel 0) de un comando se considera el teclado, y la salida estándar (*stdout* o channel 1) se considera la pantalla. También existe otro canal destinado a redirigir la salida de errores (*stderr* o channel 2) de un comando o los mensajes de error de un programa. La entrada y/o salida pueden redirigirse.

Al ejecutar un comando, a veces se desea transmitir cierta información a otro comando o redirigir la salida a un fichero específico. Cada una de estas funcionalidades se discutirá en las siguientes dos secciones.

Redirección de E/S

La redirección de E/S permite al usuario redirigir información desde o hacia un comando mediante un fichero de texto. Como se describió anteriormente, la entrada, salida y error estándares puede redirigirse, y la información puede ser tomada de los archivos de texto.

Redirección de salida estándar

Para redirigir la salida estándar a un fichero, en lugar de la pantalla, es necesario usar el operador `>` seguido del nombre del fichero. Si el fichero no existe, se creará uno nuevo; de lo contrario, el fichero existente se sobrescribirá.

Para ver el contenido del fichero recién creado, se puede usar el comando `cat`. Por defecto, este comando muestra el contenido de un fichero en la pantalla. Consulte la página del manual para obtener más información sobre sus funcionalidades.

El siguiente ejemplo muestra la funcionalidad del operador. En primera instancia, se crea un nuevo fichero que contiene el texto “Hello World!”:

```
$ echo "Hello World!" > text
$ cat text
Hello World!
```

En la segunda invocación, el mismo fichero se sobrescribe con el nuevo texto:

```
$ echo "Hello!" > text
$ cat text
Hello!
```

Si se desea agregar nueva información al final del fichero, es necesario usar el operador `>>`. Este operador también crea un nuevo fichero si no puede encontrar uno existente.

El primer ejemplo muestra la adición del texto. Como se puede ver, el nuevo texto fue agregado en la siguiente línea:

```
$ echo "Hello to you too!" >> text
$ cat text
Hello!
Hello to you too!
```

El segundo ejemplo demuestra la creación de un nuevo fichero:

```
$ echo "Hello to you too!" >> text2
$ cat text2
Hello to you too!
```

Redirección de error estándar

Para redirigir solo los mensajes de error, un usuario deberá emplear el operador `2>` seguido del nombre del archivo en el que se escribirán los errores. Si el archivo no existe, se creará uno nuevo; de lo contrario, el archivo se sobrescribirá.

Como se explicó, el canal para redirigir el error estándar es *channel 2*. Al redirigir el error estándar, se debe especificar el canal, a diferencia de la otra salida estándar donde *channel 1* está configurado por defecto. Por ejemplo, el siguiente comando busca un fichero o directorio llamado `games` y solo escribe el error en el fichero `text-error`, mientras muestra la salida estándar en la pantalla:

```
$ find /usr games 2> text-error
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
$ cat text-error
find: `games': No such file or directory
```

NOTE Para obtener más información sobre el comando `find`, consulte su página de manual.

Por ejemplo, el siguiente comando se ejecutará sin errores, por lo tanto, no se escribirá información en el fichero `text-error`:

```
$ sort /etc/passwd 2> text-error
$ cat text-error
```

Además de la salida estándar, el error estándar también se puede agregar a un fichero con el operador `2>>`. Esto agregará el nuevo error al final del fichero. Si el fichero no existe, se creará uno nuevo. El primer ejemplo muestra la adición de la nueva información en el fichero, mientras que el segundo ejemplo muestra que el comando crea un nuevo fichero porque no puede encontrar uno con el mismo nombre:

```
$ sort /etc 2>> text-error
$ cat text-error
sort: read failed: /etc: Is a directory
```

```
$ sort /etc/shadow 2>> text-error2
$ cat text-error2
sort: open failed: /etc/shadow: Permission denied
```

Usando este tipo de redirección, solo los mensajes de error serán redirigidos al fichero, la salida normal se mostrará en la pantalla o pasará por medio de la salida estándar o *stdout*.

Existe un archivo particular que técnicamente es un *bit bucket* (un archivo que acepta entrada y no hace nada con ella): `/dev/null`. Es posible redirigir cualquier información irrelevante que no se desee mostrar o redirigir a un fichero importante, como se muestra en el siguiente ejemplo:

```
$ sort /etc 2> /dev/null
```

Redirección de entrada estándar

Este tipo de redirección se utiliza para pasar datos a un comando desde un fichero específico en lugar del teclado. En este caso, el operador `<` se usa como se muestra en el ejemplo:

```
$ cat < text
Hello!
Hello to you too!
```

Generalmente la entrada estándar de redireccionamiento se usa con comandos que no aceptan argumentos de archivo. El comando `tr` es uno de ellos. Este comando se puede usar para traducir el contenido del archivo modificando los caracteres de un archivo de formas específicas; como eliminar cualquier carácter particular, por ejemplo:

```
$ tr -d "l" < text
Heo!
Heo to you too!
```

Para obtener más información, consulte la página de manual de `tr`.

Here documents

A diferencia de las redirecciones de salida, el operador `<<` actúa de manera diferente en comparación con los otros operadores. Este flujo de entrada también se denomina *Here Documents*. *Here Documents* representa el bloque de código o texto que se puede redirigir al comando o al programa interactivo. Varios tipos de lenguajes de secuencias de comandos, como `bash`, `sh` y `csh` pueden recibir información directamente desde la línea de comandos, sin utilizar ningún archivo de texto.

Como se puede ver en el siguiente ejemplo, el operador es usado para pasar datos al comando, mientras que siguiente argumento no especifica el nombre del fichero. Este argumento se interpreta como el delimitador de la entrada y no se tendrá en cuenta como contenido, por lo que `cat` no la mostrará:

```
$ cat << hello
> hey
> ola
> hello
hey
ola
```

Consulte la página del manual de `cat` para encontrar más información.

Combinaciones

La primera combinación que se tratará combina la redirección de la salida estándar y la salida de error estándar al mismo archivo. Se utilizan los operadores `&>` y `&>>`, que representan la combinación de *channel 1* y *channel 2*. El primer operador sobrescribirá el contenido existente del fichero y el segundo agregará la nueva información al final del fichero. Ambos operadores permitirán la creación del nuevo fichero si no existe, al igual que en las secciones anteriores:

```
$ find /usr admin &> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
$ find /etc/calendar &>> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
---------Omitted output----------
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
/etc/calendar
/etc/calendar/default
```

Echemos un vistazo a un ejemplo usando el comando `cut`:

```
$ cut -f 3 -d "/" newfile
$ cat newfile

share
share
share
---------Omitted output----------
lib
games
find: `admin': No such file or directory
calendar
calendar
find: `admin': No such file or directory
```

El comando `cut` corta los campos especificados del archivo de entrada usando la opción `-f`; en nuestro caso, el tercer campo . Para que el comando encuentre el campo, se debe especificar un delimitador con la opción `-d`. En nuestro caso el delimitador será el carácter `/`.

Para obtener más información sobre el comando `cut`, consulte su página de manual.

Tuberías (Pipes)

La redirección se usa principalmente para almacenar el resultado de un comando para ser procesado por otro. Este tipo de proceso puede volverse muy tedioso y complicado si se desea que los datos pasen por múltiples procesos. Para evitar esto, se puede vincular el comando directamente a través de *tuberías* o *pipes*. En otras palabras, la salida del primer comando se convierte automáticamente en la entrada del segundo. Esta conexión se realiza utilizando el operador `|` (barra vertical):

```
$ cat /etc/passwd | less
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
:
```

En el ejemplo anterior, el comando `less` después del operador `|` modifica la forma en que se muestra el archivo. El comando `less` muestra el contenido del fichero de texto permitiendo al usuario desplazarse hacia arriba y hacia abajo. `less` también se usa por defecto para mostrar las páginas de manual, como se trató en las lecciones anteriores.

Es posible usar múltiples tuberías al mismo tiempo. Los comandos intermedios que reciben entrada y luego la cambian y producen salida se denominan *filtros*. Se utilizará el comando `ls -l` para intentar contar el número de palabras de las primeras 10 líneas de la salida. Para hacer esto, es necesario usar el comando `head` que por defecto muestra las primeras 10 líneas de un archivo y luego cuenta las palabras a través del comando `wc`:

```
$ ls -l | head | wc -w
10
```

Como se mencionó anteriormente, por defecto, `head` solo imprime en pantalla las primeras 10 líneas del fichero de texto especificado. Este comportamiento puede modificarse mediante el uso de opciones específicas. Consulte la página del manual de `head` para más información.

Existe otro comando que muestra el final de un archivo: `tail`. Por defecto, este comando selecciona las últimas 10 líneas y las imprime en pantalla, y al igual que `head` el número se puede modificar. Consulte la página de manual de `tail` para más detalles.

> **NOTE** La opción `-f` puede mostrar las últimas líneas de un fichero mientras se actualiza. Esta característica puede ser muy útil cuando se monitorea continuamente un fichero como `syslog`.

Por defecto, el comando `wc` (Word count) cuenta las líneas, palabras y bytes de un archivo. Como se muestra en el ejercicio, la opción `-w` hace que el comando solo cuente las palabras dentro de las líneas seleccionadas. Las opciones más comunes que puede usar con este comando son: `-l`, que solo cuenta las líneas, y `-c`, que se usa para contar solo los bytes. Se pueden encontrar más variaciones y opciones del comando, así como más información sobre `wc` en la página del comando man.

Ejercicios guiados

1. Liste el contenido del directorio actual, incluyendo la propiedad y los permisos, y redirija la salida a un fichero llamado `contents.txt` dentro del directorio home del usuario.

2. Ordene el contenido del archivo `contents.txt` de su directorio actual y añádalo al final de un nuevo archivo llamado `contents-sorted.txt`.

3. Muestre las últimas 10 líneas del fichero `/etc/passwd` y rediríjalas a un nuevo fichero en el directorio `Documents` de su usuario.

4. Cuente el número de palabras dentro del fichero `contents.txt` y agregue la salida al final del fichero `field2.txt` en su directorio home. Deberá utilizar la redirección de entrada y salida.

5. Muestre las primeras 5 líneas del fichero `/etc/passwd` y ordene la salida en orden alfabético.

6. Usando el fichero `contents.txt` creado anteriormente, cuente el número de caracteres de las últimas 9 líneas.

7. Cuente la cantidad de ficheros llamados `test` dentro del directorio `/usr/share` y sus subdirectorios. Nota: cada salida de línea del comando `find` representa un fichero.

Ejercicios exploratorios

1. Seleccione el segundo campo del fichero `contents.txt` y redirija la salida estándar y la salida de error a otro fichero llamado `field1.txt`.

2. Usando el operador de redirección de entrada y el comando `tr`, elimine los guiones (`-`) del fichero `contents.txt`.

3. ¿Cuál es la mayor ventaja de solo redirigir errores a un fichero?

4. Reemplace todos los espacios recurrentes por un solo espacio dentro del fichero `contenidos.txt` ordenado alfabéticamente.

5. En una sola línea de comandos, elimine los espacios recurrentes (de la misma forma que en el ejercicio anterior), seleccione el noveno campo y ordénelo alfabéticamente sin distinción entre mayúsculas y minúsculas. ¿Cuántos pipes necesitó?

Resumen

En esta lección, usted aprendió:

- Tipos de redirección
- ¿Cómo usar los operadores de redirección?
- ¿Cómo usar tuberías para filtrar la salida de comandos?

Comandos usados en esta lección:

`cut`
: Elimina secciones de cada línea de un fichero.

`cat`
: Muestra o concatena ficheros.

`find`
: Busca ficheros en una jerarquía de directorios.

`less`
: Muestra el contenido de un fichero, lo que permite al usuario desplazarse línea a línea.

`more`
: Muestra el contenido de un fichero, página a página.

`head`
: Muestra las primeras 10 líneas de un fichero.

`tail`
: Muestra las últimas 10 líneas de un fichero.

`sort`
: Ordena ficheros.

`wc`
: Cuenta de forma predeterminada las líneas, palabras o bytes de un fichero.

Respuestas a los ejercicios guiados

1. Liste el contenido del directorio actual, incluyendo la propiedad y los permisos, y redirija la salida a un fichero llamado `contents.txt` dentro del directorio home del usuario.

```
$ ls -l > contents.txt
```

2. Ordene el contenido del archivo `contents.txt` de su directorio actual y añádalo al final de un nuevo archivo llamado `contents-sorted.txt`.

```
$ sort contents.txt >> contents-sorted.txt
```

3. Muestre las últimas 10 líneas del fichero `/etc/passwd` y rediríjalas a un nuevo fichero en el directorio `Documents` de su usuario.

```
$ tail /etc/passwd > Documents/newfile
```

4. Cuente el número de palabras dentro del archivo `contents.txt` y agregue la salida al final de un archivo `field2.txt` en su directorio de inicio. Deberá utilizar la redirección de entrada y salida.

```
$ wc < contents.txt >> field2.txt
```

5. Muestre las primeras 5 líneas del fichero `/etc/passwd` y ordene la salida en orden alfabético.

```
$ head -n 5 /etc/passwd | sort -r
```

6. Usando el archivo `contents.txt` creado anteriormente, cuente el número de caracteres de las últimas 9 líneas.

```
$ tail -n 9 contents.txt | wc -c
531
```

7. Cuente la cantidad de ficheros llamados `test` dentro del directorio `/usr/share` y sus subdirectorios. Nota: cada salida de línea del comando `find` representa un fichero.

```
$ find /usr/share -name test | wc -l
125
```

Respuestas a los ejercicios exploratorios

1. Seleccione el segundo campo del fichero `contents.txt` y redirija la salida estándar y la salida de error a otro fichero llamado `field1.txt`.

   ```
   $ cut -f 2 -d " " contents.txt &> field1.txt
   ```

2. Usando el operador de redirección de entrada y el comando `tr`, elimine los guiones (`-`) del fichero `contents.txt`.

   ```
   $ tr -d "-" < contents.txt
   ```

3. ¿Cuál es la mayor ventaja de solo redirigir errores a un fichero?

   La redirección de errores a un fichero puede ayudar a mantener un fichero de registro que se supervise con frecuencia.

4. Reemplace todos los espacios recurrentes por un solo espacio dentro del fichero `contenidos.txt` ordenado alfabéticamente.

   ```
   $ sort contents.txt | tr -s " "
   ```

5. En una línea de comando, elimine los espacios recurrentes (como se hizo en el ejercicio anterior), seleccione el noveno campo y ordénelo alfabéticamente y sin distinción entre mayúsculas y minúsculas. ¿Cuántas "pipes" tuviste que usar?

   ```
   $ cat contents.txt | tr -s " " | cut -f 9 -d " " | sort -fr
   ```

   El ejercicio usa 3 "pipes", uno para cada filtro.



3.2 Lección 2

Certificación:	Linux Essentials
Versión:	1.6
Tema:	3 El poder de la línea de comandos
Objetivo:	3.2 Buscar y extraer datos de los ficheros
Lección:	2 de 2

Introducción

En esta lección se tratarán algunas herramientas que se utilizan para manipular texto. Estas herramientas son usadas con frecuencia por administradores de sistemas o programas para monitorear o identificar automáticamente información específica recurrente.

Búsquedas en archivos con `grep`

La primera herramienta de esta lección es el comando `grep`. Es la abreviatura de "global regular expression print" y su principal funcionalidad es buscar patrones específicos en los ficheros. El comando muestra la línea que contiene el patrón destacado en color rojo.

```
$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1001:1001:User,,,,:/home/user:/bin/bash
```

El comando `grep`, como la mayoría de los comandos, también puede modificar su comportamiento usando varias opciones. A continuación se muestran las más comunes:

`-i`
: la búsqueda no distingue mayúsculas o minúsculas

`-r`
: la búsqueda es recursiva (busca en todos los ficheros dentro de un directorio específico y sus subdirectorios)

`-c`
: la búsqueda cuenta el número de coincidencias

`-v`
: invertir la coincidencia para imprimir líneas que no concuerde con el término de búsqueda

`-E`
: activa expresiones regulares extendidas (necesarias para algunos de los meta-caracteres más avanzados como `|`, `+` y `?`)

Consulte la página de manual para obtener más información sobre este tema.

Expresiones regulares

Otra herramienta muy poderosa y que se utiliza para describir bits de texto dentro de los ficheros, son las llamadas *expresiones regulares*. Las expresiones regulares son extremadamente útiles en la extracción de datos de ficheros de texto mediante la construcción de patrones. Se utilizan generalmente dentro de scripts o al utilizar lenguajes de alto nivel, como Perl o Python.

Al trabajar con expresiones regulares, es muy importante saber que *cada caracter cuenta* y el patrón se escribe con el propósito de que coincida con una secuencia específica de caracteres, conocida como cadena. La mayoría de los patrones utilizan los símbolos ASCII normales, como letras, dígitos, puntuación u otros símbolos, pero también se pueden utilizar caracteres Unicode para que coincidan con cualquier otro tipo de texto.

La siguiente lista explica los meta-caracteres de las expresiones regulares que se usan para formar los patrones.

`` `. ``
: Encuentra cualquier caracter individual (exceptuando saltos de línea)

`[abcABC]`
: Encuentra cualquiera de los caracteres eliminar entre corchetes

`[^abcABC]`

Encuentra cualquiera de los caracteres que no esté entre corchetes.

`[a-z]`

Encuentra cualquier caracter que esté en el rango

`[^a-z]`

Encuentra cualquier caracter que no esté en el rango

`sun|sun|moon`

Encuentra cualquiera de las cadenas de la lista

`^`

Inicio de una línea

`$`

Fin de línea

Todas las funcionalidades de las expresiones regulares se pueden implementar con `grep`. Se puede observar que en el ejemplo anterior, la palabra no está rodeada de comillas dobles. Para evitar que el shell interprete el meta-carácter en sí, se recomienda que el patrón más complejo esté entre comillas dobles (" "). Para fines prácticos, se utilizarán comillas dobles al implementar expresiones regulares, mientras que las otras comillas mantienen su funcionalidad normal, tal como se discutió en lecciones anteriores.

Los siguientes ejemplos enfatizan la funcionalidad de las expresiones regulares. Se necesitarán datos dentro del fichero, por lo tanto, el siguiente conjunto de comandos sólo agrega diferentes cadenas de texto al fichero `text.txt`.

```
$ echo "aaabbb1" > text.txt
$ echo "abab2" >> text.txt
$ echo "noone2" >> text.txt
$ echo "class1" >> text.txt
$ echo "alien2" >> text.txt
$ cat text.txt
aaabbb1
abab2
noone2
class1
alien2
```

El primer ejemplo es una combinación de búsqueda a través del fichero sin y con expresiones regulares. Para entender completamente las expresiones regulares es muy importante mostrar la diferencia. El primer comando busca la cadena exacta, en cualquier parte de la línea, mientras que el segundo busca conjuntos de caracteres que contengan cualquiera de los caracteres entre corchetes, por lo que los resultados de los comandos son diferentes.

```
$ grep "ab" text.txt
aaabbb1
abab2
$ grep "[ab]" text.txt
aaabbb1
abab2
class1
alien2
```

El segundo grupo de ejemplos muestran el uso de los metacaracteres inici y fin de línea. Es muy importante especificar la necesidad de colocar los 2 caracteres en el lugar correcto de la expresión. Cuando se especifica el comienzo de la línea, el metacaracter debe estar antes de la expresión, mientras que cuando se especifica el final de la línea, el metacaracter debe estar después de la expresión.

```
$ grep "^a" text.txt
aaabbb1
abab2
alien2
$ grep "2$" text.txt
abab2
noone2
alien2
```

Además de los metacaracteres explicados anteriormente, las expresiones regulares también incluyen metacaracteres que permiten la multiplicación del patrón previamente especificado:

`*`
: Cero o más del patrón precedente

`+`
: Uno o más del patrón precedente

`?`
: Cero o uno del patrón precedente

Para los metacaracteres, el siguiente comando busca una cadena que contenga `ab`, un solo carácter y uno o más de los caracteres encontrados previamente. El resultado muestra que `grep` encontró la cadena `aaabbb1`, que coincide con `abbb` y `abab2`. Dado que el carácter `+` es un carácter de expresión regular *extendida*, necesitamos pasar la opción `-E` al comando `grep`.

```
$ grep -E "ab.+" text.txt
aaabbb1
abab2
```

La mayoría de los metacaracteres son muy fáciles de entender, pero pueden volverse complicados cuando se usan por primera vez. Los ejemplos anteriores representan una pequeña parte de la funcionalidad de las expresiones regulares. Intente utilizar todos los metacaracteres de la tabla anterior para entender sus funcionalidades.

Ejercicios guiados

Usando `grep` y el fichero `/usr/share/hunspell/en_US.dic`, busque las líneas que coincidan con los siguientes criterios:

1. Todas las líneas que contengan la palabra `cat` en cualquier parte de la línea

2. Todas las líneas que no contengan ninguno de los siguientes caracteres: `sawgtfixkgts`.

3. Todas las líneas que comiencen con 3 letras cualquieras y la palabra `dig`.

4. Todas las líneas que terminen al menos en una `e`.

5. Todas las líneas que contengan una de las siguientes palabras: `org`, `kay` o `tuna`.

6. Cantidad de líneas que comiencen con una o ninguna `c` seguida de la cadena `ati`.

Ejercicios exploratorios

1. Construya una expresión regular que encuentre las palabras que coincidan con las de la línea "Include" y que no coincidan con las de la línea "Exclude":
   - Include: `pot`, `spot`, `apot`

     Exclude: `potic`, `spots`, `potatoe`

   - Include: `arp99`, `apple`, `zipper`

     Exclude: `zoo`, `arive`, `attack`

   - Include: `arcane`, `capper`, `zoology`

     Exclude: `air`, `coper`, `zoloc`

   - Include: `0th/pt`, `3th/tc`, `9th/pt`

     Exclude: `0/nm`, `3/nm`, `9/nm`

   - Include: `Hawaii`, `Dario`, `Ramiro`

     Exclude: `hawaii`, `Ian`, `Alice`

2. ¿Qué otro comando puede utilizarse para hacer búsqueas en ficheros? ¿Qué otras funcionalidades incluye?
3. Utilizando lo aprendido en lecciones anteriores, utilice uno de los ejemplos e intente buscar un patrón específico dentro de la salida del comando con la ayuda de `grep`.

Resumen

En esta lección, usted aprendió:

- Metacaracteres de expresiones regulares
- ¿Cómo crear patrones utilizando expresiones regulares?
- ¿Cómo hacer búsquedas en ficheros?

Comandos utilizados en los ejercicios:

`grep`
: Busca caracteres o cadenas dentro de un fichero

Respuestas a los ejercicios guiados

Usando `grep` y el fichero `/usr/share/hunspell/en_US.dic`, busque las líneas que coincidan con los siguientes criterios:

1. Todas las líneas que contengan la palabra `cat` en cualquier parte de la línea

```
$ grep "cat" /usr/share/hunspell/en_US.dic
Alcatraz/M
Decatur/M
Hecate/M
...
```

2. Todas las líneas que no contengan ninguno de los siguientes caracteres: `sawgtfixkgts`.

```
$ grep -v "[sawgtfixk]" /usr/share/hunspell/en_US.dic
49269
0/nm
1/n1
2/nm
2nd/p
3/nm
3rd/p
4/nm
5/nm
6/nm
7/nm
8/nm
...
```

3. Todas las líneas que comiencen con 3 letras cualquieras y la palabra `dig`.

```
$ grep "^...dig" /usr/share/hunspell/en_US.dic
cardigan/SM
condign
predigest/GDS
...
```

4. Todas las líneas que terminen al menos en una `e`.

```
$ grep -E "e+$" /usr/share/hunspell/en_US.dic
Anglicize
Anglophobe
Anthropocene
...
```

5. Todas las líneas que contengan una de las siguientes palabras: `org`, `kay` o `tuna`.

```
$ grep -E "org|kay|tuna" /usr/share/hunspell/en_US.dic
Borg/SM
George/MS
Tokay/M
fortunate/UY
...
```

6. Cantidad de líneas que comiencen con una o ninguna `c` seguida de la cadena `ati`.

```
$ grep -cE "^c?ati" /usr/share/hunspell/en_US.dic
3
```


Respuestas a los ejercicios exploratorios

1. Construya una expresión regular que encuentre las palabras que coincidan con las de la línea “Include” y que no coincidan con las de la línea “Exclude”:

   - Include: `pot`, `spot`, `apot`

     Exclude: `potic`, `spots`, `potatoe`

     Respuesta: `pot$`

   - Include: `arp99`, `apple`, `zipper`

     Exclude: `zoo`, `arive`, `attack`

     Respuesta: `p+`

   - Include: `arcane`, `capper`, `zoology`

     Exclude: `air`, `coper`, `zoloc`

     Respuesta: `arc|cap|zoo`

   - Include: `0th/pt`, `3th/tc`, `9th/pt`

     Exclude: `0/nm`, `3/nm`, `9/nm`

     Respuesta: `[0-9]th.+`

   - Include: `Hawaii`, `Dario`, `Ramiro`

     Exclude: `hawaii`, `Ian`, `Alice`

     Respuesta: `^[A-Z]a.*i+`

2. ¿Qué otro comando puede utilizarse para hacer búsqueas en ficheros? ¿Qué otras funcionalidades incluye?

   El comando `sed` puede buscar y reemplazar caracteres o conjuntos de caracteres dentro de un fichero.

3. Pensando en la lección anterior, use uno de los ejemplos e intente buscar un patrón específico dentro de la salida del comando, con la ayuda de `grep`.

Tomé una de las respuestas de los Ejercicios exploratorios y busqué la línea que ha leído, escrito y ejecutado como permisos de grupo. Su respuesta puede ser diferente, dependiendo del comando que eligió y el patrón que creó.

```
$ cat contents.txt | tr -s " " | grep "^....rwx"
```

Este ejercicio es para mostrarle que `grep` también puede recibir información de diferentes comandos y puede ayudarlo a filtrar la información generada.



3.3 Crear un script a partir de una serie de comandos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 3.3

Importancia

4

Áreas de conocimiento clave

- Crear scripts sencillos
- Conocer los editores de texto más comunes (vi y nano)

Lista parcial de archivos, términos y utilidades

- `#!` (shebang)
- `/bin/bash`
- Variables
- Argumentos
- Bucles `for`
- `echo`
- Estado de salida (Exit status)



3.3 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	3 El poder de la línea de comandos
Objetivo:	3.3 Convertiendo comandos en un script
Lección:	1 de 2

Introducción

Hasta ahora hemos aprendido a ejecutar comandos desde la shell, pero también podemos introducir comandos en un archivo, y luego establecer que ese archivo sea ejecutable. Cuando este se ejecuta, los comandos también se ejecutarán uno tras otro. Estos archivos se llama *scripts*, y son una herramienta absolutamente crucial para cualquier administrador de sistemas Linux.

Imprimiendo salidas (Printing Output)

Comencemos demostrando un comando que puede haber visto en lecciones anteriores: `echo`, el cual imprime un argumento en la salida estándar.

```
$ echo "Hello World!"
Hello World!
```

Ahora, usaremos la redirección de archivos para enviar este comando a un nuevo archivo llamado `new_script`.

```
$ echo 'echo "Hello World!"' > new_script
$ cat new_script
echo "Hello World!"
```

El archivo `new_script` contiene ahora el mismo comando que antes.

Cómo ejecutar un script

Vamos a demostrar algunos de los pasos necesarios para que este archivo se ejecute de la forma en que esperamos, el primer pensamiento de un usuario podría ser simplemente escribir el nombre del script. La forma en que ellos podrían escribir el nombre de cualquier otro comando:

```
$ new_script
/bin/bash: new_script: command not found
```

Podemos asumir con seguridad que `new_script` existe en nuestra ubicación actual, pero note que el mensaje de error no nos está diciendo que el *archivo* no existe, sino que nos está diciendo que el *comando* no existe, sería útil discutir cómo maneja Linux los comandos y los ejecutables.

Comandos y `PATH`

Cuando escribimos el comando `ls` en la shell, por ejemplo, estamos ejecutando un archivo llamado `ls` que existe en nuestro sistema de archivos:

```
$ which ls
/bin/ls
```

Rápidamente se volvería tedioso escribir la ruta absoluta de `ls` cada vez que deseemos ver el contenido de un directorio, así que Bash tiene una variable de entorno *(environment)* que contiene todos los directorios en los que podemos encontrar los comandos que deseamos ejecutar.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/
games:/snap/bin
```

Cada una de estas ubicaciones es donde el shell espera encontrar un comando, delimitado con dos puntos (`:`). Notará que `/bin` está presente, pero es seguro asumir que nuestra ubicación actual no lo está. La shell buscará `new_script` en cada uno de estos directorios, pero no lo encontrará y por lo

tanto lanzará el error que vimos arriba.

Hay tres soluciones para este problema: podemos mover `new_script` a uno de los directorios `PATH`, podemos añadir nuestro directorio actual a `PATH`, o podemos cambiar la forma en que intentamos llamar al script, esta última solución es la más sencilla, simplemente requiere que especifiquemos la *ubicación actual* cuando llamemos al script usando la barra de puntos (`./`).

```
$ ./new_script
/bin/bash: ./new_script: Permission denied
```

El mensaje de error ha cambiado, lo que indica que hemos hecho algunos progresos.

Ejecutar Permisos

La primera indagación que un usuario debe hacer en este caso es usar `ls -l` para mirar el archivo:

```
$ ls -l new_script
-rw-rw-r-- 1 user user 20 Apr 30 12:12 new_script
```

Podemos ver que los permisos para este archivo están configurados por defecto en `664`, pero todavía no lo hemos configurado este para que tenga *permisos de ejecución*.

```
$ chmod +x new_script
$ ls -l new_script
-rwxrwxr-x 1 user user 20 Apr 30 12:12 new_script
```

Este comando ha dado permisos de ejecución a *todos* los usuarios, tenga en cuenta que esto puede ser un riesgo para la seguridad, pero por ahora es un nivel aceptable de permisos.

```
$ ./new_script
Hello World!
```

Ahora podemos ejecutar nuestro script.

Definición del intérprete

Como hemos demostrado, hemos podido simplemente introducir texto en un archivo, configurarlo como ejecutable y ejecutarlo. El `new_script` es funcionalmente un archivo de texto normal, pero

logramos que sea interpretado por Bash, pero ¿qué pasa si está escrito en Perl o Python?

Es muy recomendable especificar el tipo de intérprete que queremos usar en la primera línea de un script, este se llama *bang line* o más conocido como *shebang*; e indica al sistema cómo queremos que se ejecute este archivo. Como estamos aprendiendo Bash, estaremos usando la ruta absoluta a nuestro ejecutable Bash, una vez más usando `which`:

```
$ which bash
/bin/bash
```

Nuestro shebang comienza con un signo de hash y un signo de exclamación, seguido de la ruta absoluta de arriba. Ahora abramos `new_script` en un editor de texto e insertemos el shebang, y aprovechemos la oportunidad para insertar un *comentario* en nuestro script, ya que los comentarios son ignorados por el intérprete y están escritos para el beneficio de otros usuarios que deseen entender su script.

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica documentar todos
los scripts.

echo "Hello World!"
```

También haremos un cambio adicional al nombre del archivo: guardaremos este archivo como `new_script.sh`. El sufijo del archivo `.sh` no cambia la ejecución del archivo de ninguna manera, es una convención que los scripts bash sean etiquetados con `.sh` o `.bash` para identificarlos fácilmente, así como los scripts Python son usualmente identificados con el sufijo `.py`.

Editores de texto comunes

Los usuarios de Linux a menudo tienen que trabajar en un entorno en el que los editores gráficos de texto no están disponibles, por lo que es muy recomendable que se familiaricen con la edición de archivos de texto desde la línea de comandos, ya que dos de los editores de texto más comunes son `vi` y `nano`.

vi

`vi` es un venerable editor de texto y está instalado por defecto en casi todos los sistemas Linux existentes. `vi` generó un clon llamado *vi IMproved* o `vim` que añade alguna funcionalidad, pero mantiene la interfaz de `vi`. Aunque trabajar con `vi` es desalentador para un nuevo usuario, el editor

es muy popular y querido por los usuarios que aprenden de sus muchas características.

La diferencia más importante entre `vi` y aplicaciones como el Bloc de notas es que `vi` tiene tres modos diferentes: Al inicio, las teclas `H`, `J`, `K` y `L` se utilizan para navegar, no para escribir. En *modo navegación*, puede pulsar `I` para entrar en *modo insertar*. En este punto, puede escribir con normalidad. Para salir de *modo insertar*, pulse `Esc` para volver a *modo navegación*.

Aunque `vi` tiene una curva de aprendizaje, los diferentes modos pueden permitir con el tiempo que un usuario experto sea más eficiente que con otros editores.

nano

`nano` es una herramienta más nueva, construida para ser simple y fácil de usar que `vi`. `nano` no tiene diferentes modos, sino que un usuario al iniciar puede empezar a escribir, y usa `Ctrl` para acceder a las herramientas impresas en la parte inferior de la pantalla.

```
                          [ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
M-E Redo
```

Los editores de texto son una cuestión de preferencia personal, y el editor que usted elija no tendrá nada que ver con esta lección, pero familiarizarse y sentirse cómodo con uno o más editores de texto valdrá la pena en el futuro.

Variables

Las variables son una parte importante de cualquier lenguaje de programación y Bash no es diferente, cuando se inicia una nueva sesión desde la terminal, la shell ya establece algunas variables, como por ejemplo la variable `PATH`, a la que llamamos variables de entorno, ya que suelen definir las características de nuestro entorno de shell, por lo que se pueden modificar y añadir variables de entorno, pero por ahora vamos a centrarnos en la configuración de variables dentro de nuestro script.

Modificaremos nuestro script para que se vea así:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username=Carol

echo "Hello $username!"
```

En este caso, hemos creado una *variable* llamada `username` y le hemos asignado el *valor* de `Carol`. Tenga en cuenta que no hay espacios entre el nombre de la variable, el signo igual o el valor asignado.

En la siguiente línea, hemos utilizado el comando `echo` con la variable, pero hay un signo de dólar (`$`) adelante del nombre de la variable, esto es importante, ya que indica a la shell que queremos tratar el `nombre de usuario` como una variable y no sólo como una palabra normal. Al introducir `$username` en nuestro comando, indicamos que queremos realizar una *sustitución*, reemplazando el *nombre* de una variable por el *valor* asignado a esa variable.

Ejecutando el nuevo script, obtenemos esta salida:

```
$ ./new_script.sh
Hello Carol!
```

- Los nombres de las variables deben contener sólo caracteres alfanuméricos o guiones bajos, y distinguen entre mayúsculas y minúsculas. `Username` y `username` serán tratadas como variables diferentes.
- La sustitución de variables también puede tener el formato `${username}`, con la adición de `{ }`. Esto también es aceptable.
- Las variables en Bash tienen un tipo *implícito*, y son consideradas como cadenas, lo que significa que la ejecución de funciones matemáticas en Bash es más complicada de lo que lo sería en otros lenguajes de programación como C/C++:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username=Carol
x=2
y=4
z=$x+$y
echo "Hello $username!"
echo "$x + $y"
echo "$z"
```

```
$ ./new_script.sh
Hello Carol!
2 + 4
2+4
```

Uso de comillas con variables

Hagamos el siguiente cambio en el valor de nuestra variable `username`:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username=Carol Smith

echo "Hello $username!"
```

Ejecutar este script nos dará un error:

```
$ ./new_script.sh
./new_script.sh: line 5: Smith: command not found
Hello !
```

Ten en cuenta que Bash es un intérprete, y como tal *interpreta* nuestro script línea por línea, en este caso, interpreta correctamente `username=Carol` para estar estableciendo una variable `username` con el

valor `Carol`, pero luego interpreta el espacio indicando el final de esa asignación, y `Smith` como el nombre de un comando.

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username="Carol Smith"

echo "Hello $username!"
```

```
$ ./new_script.sh
Hello Carol Smith!
```

Una cosa importante a tener en cuenta en Bash es que las comillas dobles y las comillas simples (`'`) se comportan de forma muy diferente, ya que se consideran "débiles" porque permiten al intérprete realizar la sustitución dentro de las comillas, mientras que las comillas simples se consideran "fuertes" porque evitan que se produzca cualquier sustitución, como en el siguiente ejemplo:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username="Carol Smith"

echo "Hello $username!"
echo 'Hello $username!'
```

```
$ ./new_script.sh
Hello Carol Smith!
Hello $username!
```

En el segundo comando `echo`, se le ha impedido al intérprete sustituir `$username` por `Carol Smith`, por lo que la salida se toma literalmente.

Argumentos

Por ejemplo, `rm testfile` contiene tanto el archivo ejecutable `rm` como el archivo de prueba `de argumento`. Los argumentos se pueden pasar al script una vez ejecutado, y modifican el comportamiento del script, por lo que son fáciles de implementar.

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username=$1

echo "Hello $username!"
```

En lugar de asignar un valor a `username` directamente dentro del script, le estamos asignando el valor de una nueva variable `$1`. Esto se refiere al valor del *primer argumento*.

```
$ ./new_script.sh Carol
Hello Carol!
```

Hay maneras de manejar más de nueve argumentos, pero eso está fuera del alcance de esta lección:

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username1=$1
username2=$2
echo "Hello $username1 and $username2!"
```

```
$ ./new_script.sh Carol Dave
Hello Carol and Dave!
```

Hay una consideración importante cuando se usan argumentos: En el ejemplo anterior, hay dos argumentos `Carol` y `Dave`, asignados a `$1` y `$2` respectivamente. Si falta el segundo argumento, por ejemplo, el intérprete de comandos no lanzará un error, el valor de `$2` será simplemente *null*, o nada en absoluto.

```
$ ./new_script.sh Carol
Hello Carol and !
```

En nuestro caso, sería una buena idea introducir algo de lógica en nuestro script para que las diferentes *condiciones* afecten a la *salida* que deseamos imprimir, comenzando por introducir otra variable útil y pasando a crear *ifs*.

Devolver el número de argumentos

Mientras que variables como `$1` y `$2` contienen el valor de los argumentos posicionales, otra variable `$#` contiene el *número de argumentos*.

```
#!/bin/bash

# Este es nuestro primer comentario. También es una buena práctica comentar todos
los scripts.

username=$1

echo "Hello $username!"
echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol Dave
Hello Carol!
Number of arguments: 2.
```

Lógica condicional

El uso de la lógica condicional en la programación es un tema vasto y no se tratará en profundidad en esta lección. Nos enfocaremos en la *sintaxis* de los condicionales en Bash, que difiere de la mayoría de los otros lenguajes de programación.

Comencemos repasando lo que esperamos conseguir, tenemos un sencillo script que debería poder imprimir un saludo a un solo usuario, y si hay algo más que un usuario, deberíamos imprimir un mensaje de error.

- La *condición* que estamos probando es el número de usuarios, que está contenido en la variable `$#`. Nos gustaría saber si el valor de `$#` es `1`.
- Si la condición es *true*, la *acción* que tomaremos es saludar al usuario.

- Si la condición es *falso*, imprimiremos un mensaje de error.

Ahora que la lógica está clara, nos centraremos en la *sintaxis* necesaria para implementar esta lógica.

```
#!/bin/bash

# A simple script to greet a single user.

if [ $# -eq 1 ]
then
  username=$1

  echo "Hello $username!"
else
  echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

La lógica condicional se encuentra entre `if` y `fi`, la condición para realizar la prueba se encuentra entre corchetes `[ ]`, y la acción a tomar en caso de que la condición sea cierta se indica después de `then`, tenga en cuenta los espacios entre los corchetes y la lógica contenida.

Este script mostrará nuestro saludo o el mensaje de error, pero siempre imprimirá la línea `Number of arguments`.

```
$ ./new_script.sh
Please enter only one argument.
Number of arguments: 0.
$ ./new_script.sh Carol
Hello Carol!
Number of arguments: 1.
```

Tome nota de la declaración `if`. Hemos usado `-eq` para hacer una comparación *numérica*. En este caso, estamos probando que el valor de `$#` es *igual* a uno. Las otras comparaciones que podemos realizar son:

`-ne`

  No igual a

`-gt`
: Mayor que

`-ge`
: Mayor que o igual a

`-lt`
: Menos que

`-le`
: Menor que o igual a

Ejercicios guiados

1. El usuario escribe lo siguiente en su shell:

```
$ PATH=~/scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH
environment variable.
ls: command not found
```

   - ¿Qué ha hecho el usuario?

   - ¿Qué comando combinará el valor actual de `PATH` con el nuevo directorio `~/scripts`?

2. Considere el siguiente script. Observe que está usando `elif` para verificar una segunda condición:

```
>  /!bin/bash

> fruit1 = Apples
> fruit2 = Oranges

  if [ $1 -lt $# ]
  then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [$1 -gt $2 ]
  then
>   echo '$fruit1 win!'
  else
>   echo "Fruit2 win!"
> done
```

   - Las líneas marcadas con un `>` contienen errores. Arregle los errores.

3. ¿Cuál será la salida en las siguientes situaciones?

```
$ ./guided1.sh 3 0
```

```
$ ./guided1.sh 2 4
```

```
$ ./guided1.sh 0 1
```

Ejercicios exploratorios

1. Escriba un script simple que verifique si se pasan exactamente dos argumentos. Si es así, imprima los argumentos en orden inverso. Considere este ejemplo (nota: su código puede verse diferente a esto, pero debería conducir a la misma salida):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

2. Este código es correcto, pero no es una comparación de números. Use una búsqueda en Internet para descubrir cómo este código es diferente de usar `-eq`.

3. Hay una variable de entorno que imprimirá el directorio actual. Use `env` para descubrir el nombre de esta variable.

4. Usando lo que has aprendido en las preguntas 2 y 3, escribe un guión corto que acepte un argumento. Si se pasa un argumento, verifique si este coincide con el nombre del directorio actual. Si es así, escriba `sí`. De lo contrario, imprima `no`.

Resumen

En esta lección usted aprendió:

- ¿Cómo crear y ejecutar scripts simples?
- ¿Cómo usar un shebang para especificar un intérprete?
- ¿Cómo establecer y usar variables dentro de scripts?
- ¿Cómo manejar argumentos en scripts?
- ¿Cómo construir declaraciones `if`?
- ¿Cómo comparar números usando operadores numéricos?

Comandos utilizados en los ejercicios:

`echo`
  Imprime una cadena a la salida estándar.

`env`
  Imprime todas las variables de entorno a la salida estándar.

`which`
  Imprime la ruta absoluta de un comando.

`chmod`
  Cambia los permisos de un archivo.

Comandos utilizados en los ejercicios:

`$1, $2, … $9`
  Contiene argumentos posicionales pasados al script.

`$#`
  Contiene el número de argumentos pasados al script.

`$PATH`
  Contiene los directorios que tienen archivos ejecutables utilizados por el sistema.

Comandos utilizados en los ejercicios:

`-ne`
: No igual a

`-gt`
: Mayor que

`-ge`
: Mayor que o igual a

`-lt`
: Menos que

`-le`
: Menor que o igual a

Respuestas a los ejercicios guiados

1. El usuario escribe lo siguiente en su shell:

```
$ PATH=~/scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH
environment variable.
ls: command not found
```

   - ¿Qué ha hecho el usuario?

     El usuario ha sobrescrito los contenidos de PATH con el directorio `~/scripts`. El comando `ls` ya no se puede encontrar, ya que no está contenido en PATH. Tenga en cuenta que este cambio solo afecta a la sesión actual, al cerrar sesión y volver a iniciar, el cambio será revertido

   - ¿Qué comando combinará el valor actual de `PATH` con el nuevo directorio `~/scripts`?

     `PATH=$PATH:~/scripts`

2. Considere el siguiente script. Observe que está usando `elif` para verificar una segunda condición:

```
>  /!bin/bash

> fruit1 = Apples
> fruit2 = Oranges

  if [ $1 -lt $# ]
  then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [$1 -gt $2 ]
  then
>   echo '$fruit1 win!'
  else
>   echo "Fruit2 win!"
> done
```

   - The lines marked with a `>` contain errors. Fix the errors.

```
#!/bin/bash

fruit1=Apples
fruit2=Oranges

if [ $1 -lt $# ]
then
  echo "This is like comparing $fruit1 and $fruit2!"
elif [ $1 -gt $2 ]
then
  echo "$fruit1 win!"
else
  echo "$fruit2 win!"
fi
```

3. ¿Cuál será la salida en las siguientes situaciones?

```
$ ./guided1.sh 3 0
```

`Apples win!`

```
$ ./guided1.sh 2 4
```

`Oranges win!`

```
$ ./guided1.sh 0 1
```

`This is like comparing Apples and Oranges!`

Respuestas a los ejercicios exploratorios

1. Escriba un script simple que verifique si se pasan exactamente dos argumentos. Si es así, imprima los argumentos en orden inverso. Considere este ejemplo (nota: su código puede verse diferente a esto, pero debería conducir a la misma salida):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

```
#!/bin/bash

if [ $# -ne 2 ]
then
  echo "Error"
else
  echo "$2 $1"
fi
```

2. Este código es correcto, pero no es una comparación de números. Use una búsqueda en Internet para descubrir cómo este código es diferente de usar `-eq`.

   El uso de `==` comparará *strings*. Es decir, si los caracteres de ambas variables coinciden exactamente, entonces la condición es verdadera.

`abc == abc`	*true*
`abc == ABC`	*false*
`1 == 1`	*true*
`1+1 == 2`	*false*

   Las comparaciones de cadenas conducen a un comportamiento inesperado si está probando números.

3. Hay una variable de entorno que imprimirá el directorio actual. Use `env` para descubrir el nombre de esta variable.

   `PWD`

4. Usando lo que has aprendido en las preguntas 2 y 3, escribe un guión corto que acepte un argumento. Si se pasa un argumento, verifique si este coincide con el nombre del directorio actual. Si es así, escriba `sí`. De lo contrario, imprima `no`.

```
#!/bin/bash

if [ "$1" == "$PWD" ]
then
  echo "yes"
else
  echo "no"
fi
```



3.3 Lección 2

Certificación:	Linux Essentials
Versión:	1.6
Tema:	3 El poder de la línea de comandos
Objetivo:	3.3 Convertiendo comandos en un script
Lección:	2 de 2

Introducción

En la última sección, hemos utilizado este sencillo ejemplo para demostrar el Bash scripting:

```
#!/bin/bash

# A simple script to greet a single user.

if [ $# -eq 1 ]
then
  username=$1

  echo "Hello $username!"
else
  echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

- Todos los scripts deben comenzar con un *shebang*, que define la ruta del intérprete.

- Todos los scripts deben incluir comentarios para describir su uso.
- Este script en particular funciona con un *argumento*, que se pasa al script cuando se llama.
- Este script contiene una *instrucción if* que prueba las condiciones de una variable incorporada `$#`. Esta variable se establece en el número de argumentos.
- Si el número de argumentos pasados al script es igual a 1, entonces el valor del primer argumento se pasa a una nueva variable llamada `username` y el script imprime un saludo al usuario. De lo contrario, se muestra un mensaje de error.
- Finalmente, el script imprime de la cantidad de argumentos. Esto es útil para la depuración.

Este es un ejemplo útil para empezar a explicar algunas de las otras características de Bash scripting.

Códigos de salida

Notarás que nuestro script tiene dos estados posibles: imprime `"Hello <user>!"` o bien imprime un mensaje de error; lo que es bastante normal para muchas de nuestras utilidades principales. Esto es bastante normal para muchas de nuestras utilidades principales. Considere `cat` con el que sin duda se ha estado familiarizando mucho.

Comparemos un uso exitoso de `cat` con una situación donde este falla. Un recordatorio de que nuestro ejemplo anterior es un script llamado `new_script.sh`.

```
$ cat -n new_script.sh

     1	#!/bin/bash
     2
     3	# A simple script to greet a single user.
     4
     5	if [ $# -eq 1 ]
     6	then
     7	  username=$1
     8
     9	  echo "Hello $username!"
    10	else
    11	  echo "Please enter only one argument."
    12	fi
    13	echo "Number of arguments: $#."
```

Este comando tiene éxito, y notará que la bandera `-n` también tiene números de línea impresos. Estos son muy útiles para depurar scripts, pero por favor tenga en cuenta que *no* son parte del script.

Ahora vamos a comprobar el valor de una nueva variable incorporada `$?`. Por ahora, sólo hay que revisar la salida:

```
$ echo $?
0
```

Ahora consideremos una situación en la que `cat` fallará, primero veremos un mensaje de error, y luego comprobaremos el valor de `$?`.

```
$ cat -n dummyfile.sh
cat: dummyfile.sh: No such file or directory
$ echo $?
1
```

La explicación de este comportamiento es el siguiente: cualquier ejecución de la utilidad `cat` devolverá un *código de salida (exit code)*. Un código de salida nos dirá si el comando tuvo éxito o presentó un error. Un código de salida de *cero* indica que el comando se completó con éxito; esto es cierto para casi todos los comandos de Linux con los que trabaje. Cualquier otro código de salida indicará un error de algún tipo. El código de salida del *último comando para ejecutar* se almacenará en la variable `$?`.

Considere un script en el que podamos estar copiando archivos a una unidad de red remota, ya que hay muchas maneras en las que la tarea de copia puede haber fallado: por ejemplo, nuestra máquina local puede no estar conectada a la red, o la unidad remota puede estar llena, y al comprobar el código de salida de nuestra utilidad de copia, podemos alertar al usuario de problemas al ejecutar el script.

Es una buena práctica implementar códigos de salida, así que lo haremos ahora, tenemos dos caminos en nuestro guión, un éxito y un fracaso, usemos el cero para indicar el éxito y el otro para indicar el fracaso.

```
 1  #!/bin/bash
 2
 3  # A simple script to greet a single user.
 4
 5  if [ $# -eq 1 ]
 6  then
 7    username=$1
 8
 9    echo "Hello $username!"
10    exit 0
11  else
12    echo "Please enter only one argument."
13    exit 1
14  fi
15  echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol
Hello Carol!
$ echo $?
0
```

Note que el comando `echo` en la línea 15 fue ignorado por completo, usando `exit` terminará el script inmediatamente, por lo que esta línea nunca se encontrará.

Manejando Muchos Argumentos

Hasta ahora nuestro script sólo puede manejar un único nombre de usuario a la vez, cualquier número de argumentos además de uno causará un error, vamos a explorar cómo podemos hacer que este script sea más versátil.

El primer instinto de un usuario puede ser usar más variables posicionales como `$2`, `$3`, etc. Desafortunadamente, no podemos anticipar el número de argumentos que un usuario puede elegir usar.

Modificaremos la lógica de nuestro script, ya que al no tener ningún argumento debería causar un error, pero cualquier otro número de argumentos debería tener éxito, este nuevo script se llamará `friendly2.sh`.

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    echo "Hello $@!"
11    exit 0
12  fi
```

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol Dave Henry!
```

Hay dos variables integradas que contienen todos los argumentos pasados al script: `$@` y `$*`. En su mayor parte, ambos se comportan igual. Bash analizará sintácticamente los argumentos y separará cada argumento cuando encuentre un espacio entre ellos. En efecto, el contenido de `$@` se ve así:

0	1	2
Carol	Dave	Henry

Si está familiarizado con otros lenguajes de programación, puede reconocer este tipo de variables como un arreglo *(array)*. Las matrices en Bash pueden crearse simplemente poniendo espacio entre elementos como la variable `FILES` en el script `arraytest` como a continuación:

```
FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"
```

Contiene una lista de muchos artículos, lo que hasta ahora no es muy útil, ya que todavía no hemos introducido ninguna forma de manejar estos artículos individualmente.

Bucles (for loop)

Veamos el ejemplo `arraytest` que se muestra antes. Si recuerdas, en este ejemplo estamos especificando una matriz propia llamada `ARCHIVOS`. Lo que necesitamos es una forma de "desempaquetar" esta variable y acceder a cada valor individual, uno tras otro. Para hacer esto, utilizaremos una estructura llamada *bucles (for loop)*, que está presente en todos los lenguajes de programación. Hay dos variables a las que nos referiremos: una es el rango y la otra es para el valor

individual en el que estamos trabajando actualmente. Este es el script en su totalidad:

```
#!/bin/bash

FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"

for file in $FILES
do
  ls -lh $file
done
```

```
$ ./arraytest
lrwxrwxrwx 1 root root 10 Apr 24 11:02 /usr/sbin/accept -> cupsaccept
-rwxr-xr-x 1 root root 54K Mar 22 14:32 /usr/sbin/pwck
-rwxr-xr-x 1 root root 43K Jan 14 07:17 /usr/sbin/chroot
```

Si te refieres de nuevo al ejemplo de `friendly2.sh`, puedes ver que estamos trabajando con un rango de valores contenidos dentro de una sola variable `$@`. Para mayor claridad, llamaremos a esta última variable `username`:

```
     1  #!/bin/bash
     2
     3  # a friendly script to greet users
     4
     5  if [ $# -eq 0 ]
     6  then
     7    echo "Please enter at least one user to greet."
     8    exit 1
     9  else
    10    for username in $@
    11    do
    12      echo "Hello $username!"
    13    done
    14    exit 0
    15  fi
```

Recuerda que la variable que definas aquí puede llamarse como quieras, y que todas las líneas dentro de `do.... done` se ejecutarán una vez para cada elemento del array:

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
```

Ahora supongamos que queremos hacer que nuestra producción parezca un poco más humana, queremos que nuestro saludo sea en una sola línea.

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    echo -n "Hello $1"
11    shift
12    for username in $@
13    do
14      echo -n ", and $username"
15    done
16    echo "!"
17    exit 0
18  fi
```

Un par de notas:

- Usando `-n` con `echo` *suprimirá la nueva línea* después de imprimir, lo que significa que todos los ecos se imprimirán en la misma línea, y la nueva línea se imprimirá sólo después de `!` en la línea 16.
- El comando `shift` eliminará el primer elemento de nuestro array, por lo que esto es así:

0	1	2
Carol	Dave	Henry

Se convierte en esto:

0	1

Dave	Henry

Observemos la salida:

```
$ ./friendly2.sh Carol
Hello Carol!
$ ./friendly2.sh Carol Dave Henry
Hello Carol, and Dave, and Henry!
```

Uso de expresiones regulares para realizar la comprobación de errores

Es posible que queramos verificar todos los argumentos que el usuario está introduciendo, por ejemplo, quizás queramos asegurarnos de que todos los nombres pasados a `friendly2.sh` contienen *sólo letras*, y cualquier carácter o número especial causará un error, para ello usaremos `grep`.

Recordemos que podemos usar expresiones regulares con `grep`.

```
$ echo Animal | grep "^[A-Za-z]*$"
Animal
$ echo $?
0
```

```
$ echo 4n1ml | grep "^[A-Za-z]*$"
$ echo $?
1
```

El `^` y el `$` indican el principio y el final de la línea respectivamente, el `[A-Za-z]` indica un rango de letras, mayúsculas o minúsculas, el `*` es un *cuantificador*, y modifica nuestro rango de letras de manera que estamos haciendo coincidir el cero con muchas letras. En resumen, nuestro `grep` tendrá éxito si la entrada es *solo* letras, y fallará si fuera lo contrario.

Lo siguiente que hay que tener en cuenta es que `grep` es devolver los códigos de salida en función de si hubo una coincidencia o no, una coincidencia positiva devuelve `0`, y una no coincidencia devuelve `1`. Podemos usar esto para probar nuestros argumentos dentro de nuestro script.

```
 1  #!/bin/bash
 2
 3  # a friendly script to greet users
 4
 5  if [ $# -eq 0 ]
 6  then
 7    echo "Please enter at least one user to greet."
 8    exit 1
 9  else
10    for username in $@
11    do
12      echo $username | grep "^[A-Za-z]*$" > /dev/null
13      if [ $? -eq 1 ]
14      then
15        echo "ERROR: Names must only contains letters."
16        exit 2
17      else
18        echo "Hello $username!"
19      fi
20    done
21    exit 0
22  fi
```

En la línea 12, estamos redirigiendo la salida estándar a `/dev/null`, la cual es una forma sencilla de suprimirla. No queremos ver ninguna salida del comando `grep`, sólo queremos probar su código de salida en lo que ocurre en la línea 13. Note también que estamos usando un código de salida de `2` para indicar un argumento inválido, por lo general es una buena práctica usar diferentes códigos de salida para indicar diferentes errores; de esta manera, un usuario experto puede usar estos códigos de salida para resolver problemas.

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
$ ./friendly2.sh 42 Carol Dave Henry
ERROR: Names must only contains letters.
$ echo $?
2
```

Ejercicios guiados

1. Lea el contenido de `script1.sh` a continuación:

```
#!/bin/bash

if [ $# -lt 1 ]
then
  echo "This script requires at least 1 argument."
  exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
  echo "no cake for you!"
  exit 2
fi

echo "here's your cake!"
exit 0
```

¿Cuál es el resultado de estos comandos?

- `./script1.sh`

- `echo $?`

- `./script1.sh cake`

- `echo $?`

- `./script1.sh CAKE`

- `echo $?`

2. Lea el contenido del archivo `script2.sh`:

```
for filename in $1/*.txt
do
   cp $filename $filename.bak
done
```

   Describa el propósito de este script tal como lo entiende.

Ejercicios exploratorios

1. Cree una secuencia de comandos que tome cualquier número de argumentos del usuario e imprima solo aquellos argumentos que sean números mayores a 10.

Resumen

En esta lección usted aprendió:

- ¿Qué son los códigos de salida, que significan y como implementarlos?
- ¿Cómo verificar el código de salida de un comando?
- ¿Qué son los bucles `for` y como usarlos con matrices?
- ¿Cómo usar `grep`, expresiones regulares y códigos de salida para verificar la entrada del usuario en los scripts?

Comandos utilizados en los ejercicios:

`shift`
: Esto eliminará el primer elemento de una matriz..

Variables especiales:

`$?`
: Contiene el código de salida del último comando ejecutado.

`$@`, `$*`
: Contiene todos los argumentos pasados al script, como una matriz.

Respuestas a los ejercicios guiados

1. Lea el contenido de `script1.sh` a continuación:

```
#!/bin/bash

if [ $# -lt 1 ]
then
  echo "This script requires at least 1 argument."
  exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
  echo "no cake for you!"
  exit 2
fi

echo "here's your cake!"
exit 0
```

¿Cuál es el resultado de estos comandos?

- comando: `./script1.sh`

  Salida: `This script requires at least 1 argument.`

- comando: `echo $?`

  Salida: `1`

- comando: `./script1.sh cake`

  Salida: `no cake for you!`

- comando: `echo $?`

  Salida: `2`

- comando: `./script1.sh CAKE`

  Salida: `here's your cake!`

  - comando: `echo $?`

    Salida: `0`

2. Lea el contenido del archivo `script2.sh`:

   ```
   for filename in $1/*.txt
   do
      cp $filename $filename.bak
   done
   ```

   Describa el propósito de este script tal como lo entiende.

   Este script realizará copias de seguridad de todos los archivos que terminen con `.txt` en un subdirectorio definido en el primer argumento.

Respuestas a los ejercicios exploratorios

1. Cree una secuencia de comandos que tome cualquier número de argumentos del usuario e imprima solo aquellos argumentos que sean números mayores a 10.

```
#!/bin/bash

for i in $@
do
  echo $i | grep "^[0-9]*$" > /dev/null
  if [ $? -eq 0 ]
  then
    if [ $i -gt 10 ]
    then
      echo -n "$i "
    fi
  fi
done
echo ""
```



Tema 4: El sistema operativo Linux



4.1 La elección del sistema operativo

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 4.1

Importancia

1

Áreas de conocimiento clave

- Diferencias entre Windows, Mac y Linux
- Gestión del ciclo de vida de una distribución

Lista parcial de archivos, términos y utilidades

- Interfaz gráfica de usuario (GUI, Graphical User Interface) versus línea de comandos; configuración de escritorio
- Ciclos de mantenimiento: beta y estable



4.1 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	4 El sistema operativo Linux
Objetivo:	4.1 Eligiendo un sistema operativo
Lección:	1 de 1

Introducción

No importa si utiliza su sistema informático en casa, universidad o en una empresa, debe tomar una decisión sobre el sistema operativo que utilizará. Usted puede elegir, especialmente si es su computadora, pero también puede escoger entre los sistemas de su compañia. Como siempre, estar bien informado sobre las opciones disponibles lo ayudará a tomar una decisión responsable. En esta lección, nuestro objetivo es ayudarlo a mantenerse informado sobre las opciones de sistemas operativos que podría elegir.

¿Qué es un sistema operativo?

Uno de los primeros aspectos que debemos estar seguros antes de elegir un sistema operativo, es entender lo que significa. El sistema operativo se encuentra en el corazón de su computadora y permite que otras aplicaciones se ejecuten por medio de éste. Además, el sistema operativo contiene controladores para acceder al hardware de la computadora, como discos y particiones, pantallas, teclados, tarjetas de red, etc. A menudo abreviaremos el sistema operativo simplemente como *OS*. Hoy en día hay muchos sistemas operativos disponibles tanto para el uso de computadoras en negocios como para el hogar. Si queremos simplificar la selección disponible para nosotros podemos agrupar las selecciones de la siguiente manera:

- Sistemas operativos basados en Linux
  - Enterprise Linux
  - Consumer Linux
- UNIX
- macOS
- Windows-based Operation Systems
  - Windows Servers
  - Windows Desktops

Eligiendo una distribución de Linux

El kernel y las distribuciones Linux

Cuando se habla de distribuciones, el sistema operativo es Linux. Linux es el *kernel* y el núcleo de cada distribución. El software del kernel es mantenido por un grupo de individuos liderado por Linus Torvalds. Torvalds es empleado de un consorcio de la industria llamado The Linux Foundation donde trabaja en el kernel de Linux.

> **NOTE** El núcleo de Linux fue desarrollado por primera vez por Linus Torvalds, un estudiante de Finlandia en 1991. En 1992, fue lanzada la primera versión de Kernel bajo GNU General Public License versión 2 (GPLv2) que fue la versión 0.12.

Linux Kernel

Como hemos mencionado, todas las distribuciones de Linux ejecutan el mismo sistema operativo, Linux.

Linux Distribution

Cuando las personas hablan de Red Hat Linux o Ubuntu Linux, se refieren a la _ Distribución de Linux_. La distribución se enviará con un kernel de Linux y un entorno que hace que el kernel sea útil de manera que podamos interactuar con él. Como mínimo, necesitaríamos un shell de línea de comandos como Bash y un conjunto de comandos básicos que nos permitan acceder y administrar el sistema. A menudo, la distribución de Linux tendrá un entorno de escritorio completo como Gnome o KDE.

Aunque cada distribución ejecuta el sistema operativo Linux, las distribuciones pueden variar y varían según la versión del sistema que utilizan. Con esto queremos decir *la versión del kernel de Linux que utiliza* cuando la distribución inicia.

TIP

Si tiene acceso a una línea de comandos de Linux en este momento, puede verificar fácilmente la versión del kernel de Linux que está ejecutando leyendo el *kernel release*:

```
$ uname -r
4.15.0-1019-aws
```

Tipos de distribuciones de Linux

Puede parecer una opción obvia ejecutar siempre la última versión del kernel de Linux, pero no es tan simple como parece. Podemos categorizar vagamente las distribuciones de Linux en tres conjuntos:

- Enterprise Grade Linux Distributions
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Consumer Grade Linux Distributions
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Experimental and Hacker Linux Distributions
  - Arch
  - Gentoo

Por supuesto, esto es un subconjunto muy pequeño de posibles distribuciones, pero la importancia es la diferencia entre las distribuciones *enterprise*, *consumer* y *experimental* y por qué existe cada una.

Enterprise Grade Linux

Las distribuciones como CentOS (*Community Enterprise OS*) están diseñadas para implementarse dentro de grandes organizaciones que utilizan hardware empresarial. Las necesidades de la empresa son muy diferentes de las necesidades de las pequeñas empresas, aficionados o usuarios domésticos. Para garantizar la disponibilidad de sus servicios, los usuarios empresariales tienen

requisitos más altos con respecto a la estabilidad de su hardware y software. Por lo tanto, las distribuciones Linux empresariales tienden a incluir versiones anteriores del kernel y otro software que funcionan de manera confiable. A menudo, las distribuciones transfieren actualizaciones importantes, como correcciones de seguridad a estas versiones estables. En la mayoría de los casos, las distribuciones empresariales pueden carecer de soporte para el hardware más reciente y proporcionar versiones anteriores de paquetes de software. Sin embargo, al igual que las distribuciones de Linux para consumidores(consumer), las empresas también tienden a elegir componentes de hardware maduros y a construir sus servicios en versiones de software estables.

Consumer Grade Linux

Las distribuciones como Ubuntu están más dirigidas a pequeñas empresas o usuarios domésticos y aficionados. Como tal, también es probable que estén utilizando el hardware más reciente que se encuentra en los sistemas de consumer. Estos sistemas necesitarán los controladores más recientes para aprovechar al máximo el nuevo hardware, pero es poco probable que la madurez tanto del hardware como de los controladores satisfaga las necesidades de las empresas más grandes. Sin embargo, para el mercado de consumo, el último kernel es exactamente lo que se necesita con los controladores más modernos, incluso si están poco probados. Los nuevos núcleos de Linux tendrán los controladores más recientes para admitir el hardware más reciente que probablemente esté en uso. Especialmente con el desarrollo que vemos con Linux en el mercado de juegos, es muy importante que los últimos controladores estén disponibles para estos usuarios.

> **NOTE** Algunas distribuciones como Ubuntu proporcionan versiones de grado de consumidor que contienen software reciente y reciben actualizaciones por un período de tiempo bastante pequeño, así como las llamadas versiones de soporte a largo plazo, LTS (Long Term Support versions) para abreviar, que son más adecuadas para entornos empresariales.

Experimental and Hacker Linux Distributions

Distribuciones como Arch Linux o Gentoo Linux viven a la vanguardia de la tecnología. Contienen las versiones más recientes de software, incluso si tiene errores y características no probadas. A cambio, estas distribuciones tienden a usar un modelo de lanzamiento continuo que les permite entregar actualizaciones en cualquier momento. Estas distribuciones son utilizadas por usuarios avanzados que desean recibir siempre el software más reciente y son conscientes de que la funcionalidad puede romperse en cualquier momento y luego necesiten reparar sus sistemas.

En resumen, al considerar Linux como su sistema operativo, si está utilizando hardware de grado empresarial en sus servidores o equipos de escritorio, puede hacer uso de distribuciones de grado empresarial o de consumo. Si está utilizando hardware de calidad para el consumidor y necesita

aprovechar al máximo las últimas innovaciones de hardware, es probable que necesite una distribución de Linux similar para satisfacer las necesidades del hardware.

Algunas distribuciones de Linux están relacionadas entre sí. Ubuntu, por ejemplo, está basado en Debian Linux y utiliza el mismo sistema de empaquetado, DPKG. Fedora, como otro ejemplo, es un banco de pruebas para RedHat Enterprise Linux, donde las características potenciales de futuras versiones de RHEL se pueden explorar antes de su disponibilidad en la distribución empresarial.

Además de las distribuciones que hemos mencionado aquí hay muchas otras. Linux tiene la ventaja de que al ser software de código abierto muchas personas pueden desarrollar lo que creen que debería ser Linux. Por ende, tenemos cientos de distribuciones a elegir. Para ver más distribuciones de Linux, puede visitar The Distro Watch Web Site, los encargados del sitio web enumeran las 100 principales descargas de distribuciones de Linux, lo que le permite comparar y ver en ese momento que es lo más popular.

Ciclo de vida del soporte de Linux

Como es de esperar, las distribuciones Linux empresariales tienen una vida útil más larga que las ediciones de Linux para consumidores o comunidades. Por ejemplo, Red Hat Enterprise Linux tiene soporte durante 10 años. Red Hat Enterprise Linux 8 se lanzó en mayo de 2019, mientras que las actualizaciones de software y el soporte están disponibles hasta mayo de 2029.

Las ediciones para el cliente a menudo sólo cuentan con el apoyo de la comunidad a través de foros. Las actualizaciones de software suelen estar disponibles para 3 ediciones. Si tomamos Ubuntu como ejemplo, en el momento de escribir este artículo, la versión 19.04 era el último disponible teniendo actualizaciones hasta la versión 19.10 y detiendose en enero de 2020. Ubuntu también provee ediciones con soporte a largo plazo, conocidas como ediciones *LTS*, que tienen 5 años de soporte desde la versión original. La versión actual de LTS es la 18.04 que tendrá actualizaciones de software hasta el 2023. Estas versiones LTS hacen de Ubuntu una posible opción para la empresa con soporte comercial disponible de Canonical (la empresa que está detrás de la marca Ubuntu) o de empresas consultoras independientes.

NOTE Las distribuciones de Ubuntu usan números de versión basados en fechas en el formato AA.MM: Por ejemplo, la versión 19.04 se lanzó en abril de 2019.

Linux como su escritorio

Usar Linux como su sistema de escritorio puede ser más desafiante en una empresa donde el soporte de escritorio se centra en las ofertas comerciales de sistemas operativos. Sin embargo, no solo el soporte puede resultar desafiante. Un cliente empresarial también puede haber realizado grandes inversiones en soluciones de software que los vinculan a sistemas operativos de escritorio

específicos. Dicho esto, hay muchos ejemplos de escritorios Linux que se integran en grandes organizaciones con compañías como Amazon, incluso con su propia distribución de Linux Amazon Linux 2. Esto se usa en su plataforma en la nube de AWS, pero también internamente para servidores y equipos de escritorio.

Usar Linux en una empresa más pequeña o en casa se vuelve mucho más fácil y puede ser una experiencia gratificante, ya que elimina la necesidad de licencias y abre los ojos a la gran cantidad de software libre y de código abierto que está disponible para Linux. También encontrará que hay muchos entornos de escritorio diferentes disponibles. Los más comunes son Gnome y KDE, sin embargo, existen muchos otros. La decisión se reduce a la preferencia personal.

Usar Linux en servidores

Usar Linux como su sistema operativo de servidor es una práctica común en el sector empresarial. Los servidores son mantenidos por ingenieros especializados en Linux. Entonces, incluso con miles de usuarios, los usuarios pueden permanecer ignorantes de los servidores a los que se están conectando. El sistema operativo del servidor no es importante para ellos; en general, las aplicaciones del cliente no diferirán entre Linux y otros sistemas operativos en el backend. También es cierto que a medida que más aplicaciones se virtualizan o se colocan en contenedores dentro de nubes locales y remotas, el sistema operativo se enmascara aún más y es probable que el sistema operativo incorporado sea Linux.

Linux en la nube

Otra oportunidad para familiarizarse con Linux es implementarlo dentro de una de las muchas nubes públicas disponibles. Crear una cuenta con uno de los muchos proveedores de la nube le permitirá implementar rápidamente muchas distribuciones de Linux diferentes de forma rápida y fácil.

Sistema operativo no Linux

Por increíble que parezca, hay sistemas operativos que no están basados en el kernel de Linux. Por supuesto, a lo largo de los años han aparecido muchos y algunos se han quedado en el camino, pero aún hay otras opciones disponibles para usted. Ya sea en casa o en la oficina.

Unix

Antes de tener Linux como sistema operativo, existía Unix. Unix solía venderse junto con el hardware y todavía hoy en día hay varios Unix comerciales como AIX y HP-UX disponibles en el mercado. Si bien Linux se inspiró mucho en Unix (y la falta de disponibilidad para cierto hardware), además, la familia de sistemas operativos BSD se basa directamente en Unix. Hoy, FreeBSD, NetBSD y OpenBSD junto con algunos otros sistemas BSD relacionados, están disponibles como software libre.

Unix se utilizó mucho en la empresa, pero hemos visto una disminución en la suerte de Unix con el crecimiento de Linux. A medida que Linux ha crecido y las ofertas de soporte empresarial también han crecido, hemos visto que Unix comienza a desaparecer lentamente. Solaris, originalmente de Sun antes de mudarse a Oracle, ha desaparecido recientemente. Este fue uno de los sistemas operativos Unix más grandes utilizados por las compañías de telecomunicaciones, conocido como *Telco Grade Unix*.

Los sistemas operativos Unix incluyen:

- AIX
- FreeBSD, NetBSD, OpenBSD
- HP-UX
- Irix
- Solaris

macOS

macOS (anteriormente OS X) de Apple se remonta a 2001. Basado en BSD Unix y haciendo uso de la línea de comandos Bash, es un sistema amigable para usar si estás acostumbrado a sistemas Unix o Linux. Si está utilizando macOS, puede abrir la aplicación de la terminal para acceder a la línea de comando. Al ejecutar el mismo comando `uname` nuevamente, podemos verificar el sistema operativo informado:

```
$ uname -s
Darwin
```

NOTE En este caso, utilizamos la opción `-s` para devolver el nombre del sistema operativo. Anteriormente utilizamos `-r` para devolver el número de versión del núcleo.

Microsoft Windows

Todavía podemos decir que la mayoría de las computadoras de escritorio y portátiles estarán basadas en Windows. El sistema operativo ha sido verdaderamente exitoso y ha dominado el mercado de las computadoras de escritorio durante años. Aunque es un software patentado y no es gratuito, a menudo la licencia del sistema operativo se incluye cuando compra el hardware, por lo que se convierte en la opción más fácil de tomar. Por supuesto, existe un amplio soporte para Windows en todos los proveedores de hardware y software, así como muchas aplicaciones de código abierto

disponibles para Windows. El futuro para Windows no parece tan brillante como lo ha sido. Con menos computadoras de escritorio y portátiles vendidas, ahora el enfoque está en el mercado de tabletas y teléfonos. Esto ha sido dominado por Apple y Android y es difícil para Microsoft ganar terreno.

Como plataforma de servidor, Microsoft ahora permite a sus clientes elegir entre una GUI (*Graphical User Interface*) y una versión de línea de comando solamente. La separación de la GUI y la línea de comando es importante. La mayoría de las veces se cargará la GUI de los servidores Microsoft más antiguos, pero nadie la usará. Considere un controlador de dominio de Active Directory, los usuarios lo usan todo el tiempo para autenticarse en el dominio, pero se administra de forma remota desde los escritorios de los administradores y no desde el servidor.

Guía de ejercicios

1. ¿Qué proyecto constituye el componente común de todas las distribuciones de Linux?

CentOS	
Red Hat	
Ubuntu	
Linux Kernel	
CoreOS	

2. ¿Qué sistema operativo es utilizado hoy en día para MacOS de Apple?

OS X	
OSX	
Darwin	
MacOS	

3. ¿En qué se diferencia una distribución de Linux del kernel de Linux?

El kernel es parte de una distribución, la distribución como aplicaciones envuelven el núcleo para que sea útil	
El kernel es la distribución de Linux	
Todas las distribuciones que usan el mismo kernel son iguales	

4. ¿Cuál de los siguientes es un entorno de escritorio en Linux?

Mint	
Elementary	
Zorin	
Wayland	

5. ¿Qué componente de un sistema operativo permite el acceso al hardware?

Drivers	
Shells	
Service	
Application	

Ejercicios de exploración

1. Recupere la versión actual de Kernel de su sistema Linux si tiene acceso a la línea de comando

2. Utilizando su motor de búsqueda preferido, localice e identifique los proveedores de nube pública disponibles para usted. Estos podrían incluir AWS, Google Cloud, Rackspace y muchos más. Elija uno y revise qué sistemas operativos están disponibles para implementar.

Resumen

En esta sección, ha aprendido a diferenciar entre los diferentes sistemas operativos comúnmente disponibles. Nosotros discutimos:

- Linux Based Operating Systems
- UNIX
- macOS
- Windows Based Operation Systems

Dentro de la categoría de Linux, podríamos dividir aún más la selección en distribuciones con soporte a largo plazo y aquellas con un ciclo de soporte más corto. Las versiones LTS son más adecuadas para las empresas y el soporte a corto plazo está dirigido a usuarios domésticos y aficionados.

- Enterprise Grade Linux Distributions
  - Red Hat Enterprise Linux
  - CentOS
  - SUSE Linux Enterprise Server
  - Debian GNU/Linux
  - Ubuntu LTS
- Consumer Grade Linux Distributions
  - Fedora
  - Ubuntu non-LTS
  - openSUSE
- Experimental and Hacker Linux Distributions
  - Arch
  - Gentoo

Respuestas a los ejercicios guiados

1. ¿Qué proyecto constituye el componente común de todas las distribuciones de Linux?

CentOS	
Red Hat	
Ubuntu	
Linux Kernel	X
CoreOS	

2. ¿Qué sistema operativo es utilizado hoy en día para MacOS de Apple?

OS X	
OSX	
Darwin	X
MacOS	

3. ¿En qué se diferencia una distribución de Linux del kernel de Linux?

El kernel es parte de una distribución, la distribución como aplicaciones envuelven el núcleo para que sea útil	X
El kernel es la distribución de Linux	
Todas las distribuciones que usan el mismo kernel son iguales	

4. ¿Cuál de los siguientes es un entorno de escritorio en Linux?

Mint	
Elementary	
Zorin	
Wayland	X

5. ¿Qué componente de un sistema operativo permite el acceso al hardware?

Drivers	X
Shells	
Service	
Application	

Respuestas a los ejercicios explorativos

1. Recupere la versión actual de Kernel de su sistema Linux si tiene acceso a la línea de comando

```
$ uname -r
4.15.0-47-generic
```

2. Utilizando su motor de búsqueda preferido, localice e identifique los proveedores de nube pública disponibles para usted. Estos podrían incluir AWS, Google Cloud, Rackspace y muchos más. Elija uno y revise qué sistemas operativos están disponibles para implementar.

   Como ejemplo, AWS le permite implementar muchas distribuciones de Linux como Debian, Red Hat, SUSE o Ubuntu, así como Windows.



4.2 Conocer el hardware del ordenador

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 4.2

Importancia

2

Áreas de conocimiento clave

- Hardware

Lista parcial de archivos, términos y utilidades

- Placas base, procesadores, fuentes de alimentación, discos ópticos y periféricos
- Discos duros mecánicos, unidades de estado sólido, particiones, /dev/sd*.
- Controladores o drivers



4.2 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	4 El sistema operativo Linux
Objetivo:	4.2 Comprendiendo el hardware de la computadora
Lección:	1 de 1

Introducción

Sin hardware, el software no es más que otra forma de literatura. El hardware procesa los comandos descritos por el software y proporciona mecanismos de almacenamiento, entrada y salida. Incluso la nube está respaldada por hardware.

Como sistema operativo, una de las responsabilidades de Linux es proporcionar software con interfaces para acceder al hardware de un sistema. La mayoría de los detalles de configuración están más allá del alcance de esta lección. Sin embargo, los usuarios a menudo se preocupan por el rendimiento, la capacidad y otros factores del hardware del sistema, ya que afectan la capacidad de un sistema para soportar adecuadamente aplicaciones específicas. Esta lección analiza el hardware como elementos físicos que utilizan interfaces y conectores estándar. Los estándares son relativamente estáticos, pero el factor de forma, rendimiento y las características de capacidad del hardware están en constante evolución. Independientemente de cómo los cambios pueden difuminar las distinciones físicas, los aspectos conceptuales del hardware descritos en esta lección aún se aplican.

NOTE En varios puntos dentro de esta lección, se usan ejemplos de línea de comandos para demostrar formas de acceder a información sobre hardware. La mayoría de los ejemplos son de una Raspberry Pi B+, pero deberían aplicarse a la mayoría de los sistemas. No es necesario saber estos comandos para comprender este material.

Fuentes de alimentación

Todos los componentes activos en un sistema informático requieren electricidad para funcionar. Desafortunadamente la mayoría de las fuentes de electricidad no son apropiadas. El hardware del sistema informático requiere voltajes específicos con tolerancias relativamente ajustadas. Esto comúnmente no es lo que está disponible en su toma de corriente.

Las fuentes de alimentación normalizan las fuentes de energía disponibles. Los requisitos de voltaje estándar permiten a los fabricantes crear componentes de hardware que pueden usarse en sistemas en cualquier parte del mundo. Las fuentes de alimentación de escritorio tienden a utilizar la electricidad de los tomas corrientes de pared. Las fuentes de alimentación del servidor tienden a ser más críticas, por lo que a menudo pueden conectarse a múltiples fuentes para garantizar que continúen funcionando si falla una alguna.

El consumo de energía genera calor y el calor excesivo puede hacer que los componentes del sistema funcionen lentamente o incluso fallen. La mayoría de los sistemas tienen algún tipo de ventilador para mover el aire para una refrigeración más eficiente. Componentes como los procesadores a menudo generan calor que el flujo de aire por sí solo no puede disipar. Estos componentes calientes se unen una aleación especial conocidos como disipadores para ayudar a dispersar el calor que generan. Los disipadores de calor a menudo tienen su propio ventilador local pequeño para garantizar un flujo de aire adecuado.

Tarjeta madre

Todo el hardware de un sistema necesita interconectarse. Una placa base (Tarjeta madre) normaliza esa interconexión utilizando conectores estandarizados y factores de forma. También proporciona soporte para la configuración y las necesidades eléctricas de esos conectores.

Hay una gran cantidad de configuraciones en una placa base, estas admiten diferentes procesadores y sistemas de memoria, también tienen diferentes combinaciones de conectores estandarizados que se adaptan a los diferentes tamaños. Quizás por la capacidad de conectar dispositivos externos específicos, la configuración de la placa base es transparente para los usuarios. Sin embargo, los administradores están expuestos principalmente a la configuración de la placa base cuando es necesario identificar dispositivos específicos.

Cuando se aplica la alimentación por primera vez, hay un hardware específico de la placa base que

debe configurarse e inicializarse antes de que el sistema pueda funcionar. Las placas base utilizan una programación almacenada en la memoria no volátil conocida como firmware para manejar el hardware específico de la tarjeta. La forma original del firmware de la placa base fue conocía como BIOS (*Basic Input/Output System*). Más allá de los ajustes de configuración básica, el BIOS era el principal responsable de identificar, cargar y transferir la operación a un sistema operativo como Linux. A medida que el hardware evolucionó, el firmware se expandió para admitir discos más grandes, diagnósticos, interfaces gráficas, redes y otras capacidades avanzadas. Intel definió un estándar para firmware conocido como EFI (*Extensible Firmware Interface*). Luego Intel contribuyó con una organización de estándares para crear UEFI (*Unified Extensible Firmware Interface*). Hoy en día la mayoría de las placas base usan UEFI, sin embargo la mayoría de las personas todavía se refieren al firmware de la placa base como BIOS.

Hay muy pocas configuraciones de firmware que sea de interés para los usuarios en general, por lo que solo las personas responsables de la configuración del hardware de un sistema necesitarán lidiar con sus configuraciones. Una de las pocas opciones que comúnmente se modifican es habilitar las extensiones de virtualización de las CPU modernas.

Memoria

La memoria contiene los datos y código del programa de todas aquellas aplicaciones que se ejecutan en ese momento. Cuando hablan de la memoria de la computadora, la mayoría de las personas se refieren a la memoria de este sistema. Otro término común utilizado para la memoria del sistema es el acrónimo RAM (*Random Access Memory*) o alguna variación de ese acrónimo. En ocasiones, también se utilizan referencias al paquete físico de la memoria del sistema como DIMM, SIMM o DDR.

La memoria usualmente se conectan en módulos individuales integrados a la placa base. Los módulos de memoria individual actualmente varían en tamaño de 2 GB a 64 GB. Para la mayoría de las aplicaciones de uso general 4 GB es la memoria mínima que podrían considerar. Para estaciones de trabajo individuales, 16 GB suelen ser más que suficiente. Sin embargo, incluso 16 GB puede ser limitante para los usuarios que ejecutan juegos, video o aplicaciones de audio de alta gama. Los servidores a menudo requieren 128 GB o incluso 256 GB de memoria para soportar eficientemente las cargas de los usuarios.

En su mayor parte, Linux permite a los usuarios tratar la memoria del sistema como una caja negra. Se inicia una aplicación y Linux se encarga de asignar la memoria requerida. Linux libera la memoria para que otras aplicaciones la utilicen cuando se completa una aplicación, pero, ¿Qué sucede si una aplicación requiere más que la memoria del sistema disponible? En este caso, Linux mueve las aplicaciones inactivas de la memoria del sistema a un área de disco especial conocida como espacio de intercambio, luego mueve nuevamente las aplicaciones inactivas desde el espacio de intercambio a la memoria del sistema cuando necesiten ejecutarse.

A menudo los sistemas sin hardware de video dedicado usan una parte de la memoria del sistema (comúnmente 1 GB) para actuar como almacenamiento de video. Esto reduce la memoria efectiva del sistema. El hardware de video dedicado generalmente tiene su propia memoria separada que no está disponible como memoria del sistema.

Hay varias formas de obtener información sobre la memoria del sistema. Como usuario, la cantidad total de memoria disponible y en uso son típicamente los valores de interés. Una fuente de información sería ejecutar el comando `free` junto con el parámetro `-m` para usar megabytes en la salida:

```
$ free -m
              total        used        free      shared  buff/cache   available
Mem:            748          37          51          14         660         645
Swap:            99           0          99
```

La primera línea indica la memoria total disponible para el sistema (`total`), la memoria en uso (`used`) y la memoria libre (`free`). La segunda línea muestra esta información para el espacio de intercambio. La memoria indicada como `shared` y `buff/cache` se usa actualmente para otras funciones del sistema, aunque la cantidad indicada en `available` podría usarse para una aplicación.

Procesadores

La palabra "procesador" implica que algo está siendo procesado. En las computadoras, la mayor parte de ese procesamiento se trata de señales eléctricas. Típicamente, esas señales se tratan como si tuvieran uno de los valores binarios 1 o 0.

Cuando las personas hablan de computadoras, a menudo usan el procesador de texto de manera intercambiable con el acrónimo CPU (*Central Processing Unit*). Lo que no es técnicamente correcto. Cada computadora de uso general tiene una CPU que procesa los comandos binarios especificados por el software. Por lo tanto, es comprensible que las personas intercambien procesador y CPU. Sin embargo, además de una CPU, las computadoras modernas a menudo incluyen otros procesadores específicos de tareas. Quizás el procesador adicional más reconocible es una GPU (*Graphical Processing Unit*). Por lo tanto, mientras que una CPU es un procesador, no todos los procesadores son CPU.

Para la mayoría de las personas, la arquitectura de la CPU es una referencia a las instrucciones que admite el procesador. Aunque Intel y AMD fabrican procesadores que admiten las mismas instrucciones, es significativo diferenciar por proveedor debido a las diferencias específicas de empaque, rendimiento y consumo de energía del proveedor. Las distribuciones de software comúnmente usan estas designaciones para especificar el conjunto mínimo de instrucciones que requieren para operar:

i386

Hace referencia al conjunto de instrucciones de 32 bits asociado con el Intel 80386.

x86

Por lo general, hace referencia a los conjuntos de instrucciones de 32 bits asociados con los sucesores al 80386, como 80486, 80586 y Pentium.

x64 / x86-64

Procesadores de referencias que admiten las instrucciones de 32 bits y de 64 bits de la familia x86.

AMD

Una referencia al soporte x86 de los procesadores AMD.

AMD64

Una referencia al soporte x64 de los procesadores AMD.

ARM

Hace referencia a una CPU de *Reduced Instruction Set Computer* (RISC) que no se basa en el conjunto de instrucciones x86. Comúnmente utilizado por dispositivos embebidos, móviles, tabletas y dispositivos con batería. Raspberry Pi utiliza una versión de Linux para ARM.

El archivo `/proc/cpuinfo` contiene información detallada sobre el procesador de un sistema. Lamentablemente, los detalles no son amigables para los usuarios en general. Se puede obtener un resultado más general con el comando `lscpu`. Salida de una Raspberry Pi B+:

```
$ lscpu
Architecture:          armv7l
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
Model:                 4
Model name:            ARMv7 Processor rev 4 (v7l)
CPU max MHz:           1400.0000
CPU min MHz:           600.0000
BogoMIPS:              38.40
Flags:                 half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva
idivt vfpd32 lpae evtstrm crc32
```

Para la mayoría de las personas, la gran cantidad de proveedores, familias de procesadores y factores de especificación representan una variedad desconcertante de opciones. En cualquier caso, hay varios factores asociados con las CPU y los procesadores que incluso los usuarios y administradores generales deben tener en cuenta cuando necesitan especificar entornos operativos:

Bit size

Para las CPU, este número se relaciona tanto con el tamaño nativo de datos que manipula, así como la cantidad de memoria a la que puede acceder. La mayoría de los sistemas modernos son de 32 bits o de 64 bits. Si una aplicación necesita acceso a más de 4 gigabytes de memoria, debe ejecutarse en un sistema de 64 bits, ya que 4 gigabytes es la dirección máxima que se puede representar con 32 bits. Y, aunque las aplicaciones de 32 bits generalmente se pueden ejecutar en sistemas de 64 bits, las aplicaciones de 64 bits no se pueden ejecutar en sistemas de 32 bits.

Clock speed

A menudo se expresa en megahercios (MHz) o gigahercios (GHz). Esto se relaciona con la rapidez con que un procesador ejecuta las instrucciones. Pero la velocidad del procesador es solo uno de los factores que afectan los tiempos de respuesta, los de espera y el rendimiento del sistema. Incluso un usuario "multi-tasking" rara vez mantiene activa una CPU de una PC de escritorio por más del 2 o 3 por ciento del tiempo. En cualquier caso, si utiliza con frecuencia aplicaciones intensivas que involucran actividades como el cifrado o la representación de video, la velocidad de la CPU puede tener un impacto significativo en el rendimiento y el tiempo de espera.

Cache

Las CPU requieren un flujo constante de instrucciones y datos para funcionar. El costo y el consumo de energía de una memoria de sistema de varios gigabytes a la que se pueda acceder a velocidades del reloj podría ser prohibitiva. La memoria caché del CPU está integrada en su chip para proporcionar un búfer de alta velocidad entre las CPU y la memoria del sistema. La memoria caché se separa en varias capas, comúnmente denominadas L1, L2, L3 e incluso L4. En el caso de la memoria caché, entre mayor sea suele ser mejor.

Cores

Core o núcleo se refiere a una CPU individual, además el núcleo representa una CPU física. *Hyper-Threading Technology* (HTT) permite que una sola CPU física procese simultáneamente múltiples instrucciones actuando virtualmente como múltiples CPU físicas. Por lo general, los núcleos físicos múltiples se empaquetan como un solo chip de procesador físico. Sin embargo, hay placas base que admiten múltiples chips de procesadores físicos. En teoría, tener más núcleos para procesar tareas siempre parece producir un mejor rendimiento del sistema. Desafortunadamente, las aplicaciones de escritorio a menudo solo mantienen ocupadas las CPU 2 o 3 por ciento del tiempo, por lo que agregar más CPU inactivas probablemente resulte en una mejora mínima del rendimiento. Más núcleos son los más adecuados para ejecutar aplicaciones

que están escritas para tener múltiples subprocesos independientes de operación, como la representación de cuadros de video, la representación de páginas web o entornos de máquinas virtuales multiusuario.

Almacenamiento

Los dispositivos de almacenamiento proporcionan un método para retener programas y datos. Las unidades de disco duro (HDD) y las unidades de estado sólido (SSD) son la forma más común de dispositivos de almacenamiento para servidores y equipos de escritorio. También se utilizan dispositivos de memoria USB y dispositivos ópticos como DVD, pero rara vez como dispositivo principal.

Como su nombre lo indica, una unidad de disco duro almacena información en uno o más discos físicos. Los discos físicos están cubiertos con medios magnéticos para hacer posible el almacenamiento. Los discos están dentro de un paquete sellado ya que el polvo, las partículas pequeñas e incluso las huellas digitales interferirían con la capacidad del HDD para leer y escribir en los medios magnéticos.

Las SSD son versiones más sofisticadas de memorias USB con una capacidad significativamente mayor. Los SSD almacenan información en microchips para que no haya partes móviles.

Aunque las tecnologías subyacentes para HDD y SSD son diferentes, hay factores importantes que se pueden comparar. La capacidad de la unidad de disco duro se basa en el escalado de componentes físicos, mientras que la capacidad de la unidad de estado sólido depende del número de microchips. Por gigabyte, los SSD cuestan entre 3 y 10 veces más de lo que cuesta un HDD. Para leer o escribir, un HDD debe esperar a que una ubicación en un disco gire a una ubicación conocida, mientras que los SSD son de acceso aleatorio. Las velocidades de acceso SSD son generalmente de 3 a 5 veces más rápidas que los dispositivos HDD. Como no tienen partes móviles, los SSD consumen menos energía y son más confiables que los HDD.

La capacidad de almacenamiento aumenta constantemente para HDD y SSD. Hoy en día, 5 terabytes de HDD y 1 terabyte de SSD están comúnmente disponibles. En cualquier caso, una gran capacidad de almacenamiento no siempre es mejor. Cuando un dispositivo de almacenamiento falla, la información que contiene ya no está disponible. Y, por supuesto, la copia de seguridad lleva más tiempo cuando hay más información para hacer una copia de seguridad. Para las aplicaciones que leen y escriben muchos datos, la latencia y el rendimiento pueden ser más importantes que la capacidad.

Los sistemas modernos utilizan SCSI (*Small Computer System Interface*) o SATA (*Serial AT Attachment*) para conectarse con dispositivos de almacenamiento. Estas interfaces generalmente son compatibles con el conector apropiado en la placa base. La carga inicial proviene de un dispositivo de

almacenamiento conectado a la placa base. La configuración del firmware define el orden en el que se accede a los dispositivos para esta carga inicial.

Los sistemas de almacenamiento conocidos como RAID (*Redundant Array of Independent Disks*) son una implementación común para evitar la pérdida de información. Una matriz RAID consta de múltiples dispositivos físicos que contienen copias duplicadas de información. Si un dispositivo falla, toda la información aún está disponible. Las diferentes configuraciones físicas de RAID se mencionan como 0, 1, 5, 6 y 10. Cada designación tiene diferentes tamaños de almacenamiento, características de rendimiento y formas de almacenar datos redundantes o sumas de comprobación para la recuperación de datos. Más allá de algunos gastos generales de configuración administrativa, la existencia de RAID es efectivamente transparente para los usuarios.

Los dispositivos de almacenamiento comúnmente leen y escriben datos como bloques de bytes. El comando `lsblk` se puede usar para enumerar los dispositivos de bloques disponibles para un sistema. El siguiente ejemplo se ejecutó en una Raspberry Pi usando una tarjeta SD como dispositivo de almacenamiento. Los detalles de la salida están cubiertos en las lecciones *Particiones* y *Drivers* :

```
$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 29.7G  0 disk
+-mmcblk0p1 179:1    0 43.9M  0 part /boot
+-mmcblk0p2 179:2    0 29.7G  0 part /
```

Particiones

Un dispositivo de almacenamiento es efectivamente una larga secuencia de ubicaciones de almacenamiento. La partición es el mecanismo que le dice a Linux si debe ver estas ubicaciones de almacenamiento como una secuencia única o múltiples secuencias independientes. Cada partición se trata como si fuera un dispositivo individual. La mayoría de las veces las particiones se crean cuando un sistema se configura por primera vez. Si se necesita un cambio, hay herramientas administrativas disponibles para administrar la partición de dispositivos.

Entonces, ¿por qué serían deseables múltiples particiones? Algunos ejemplos para usar particiones serían administrar el almacenamiento disponible, aislar la sobrecarga de cifrado o admitir múltiples sistemas de archivos. Las particiones permiten tener un único dispositivo de almacenamiento que puede iniciarse en diferentes sistemas operativos.

Si bien Linux puede reconocer la secuencia de almacenamiento de un dispositivo sin formato, un dispositivo sin formato no podría usarse; para usar este dispositivo primero debe ser formateado. El formateo escribe un sistema de archivos en un dispositivo y lo prepara para las operaciones de archivo, sin un sistema de archivos un dispositivo no podrá utilizarse para operaciones relacionadas

con archivos.

Los usuarios ven las particiones como si fueran dispositivos individuales. Esto hace que sea fácil pasar por alto el hecho de que todavía están tratando con un solo dispositivo físico. En particular, las operaciones de un dispositivo en realidad son operaciones que se realizan es una partición. Un solo dispositivo es un mecanismo físico con un conjunto de hardware de lectura / escritura. Más importante aún, no puede usar particiones de un solo dispositivo físico como un diseño tolerante a fallas. Si el dispositivo falla, todas las particiones fallan, por lo que no habría tolerancia a fallas.

NOTE	*Logical Volume Manager* (LVM) es una capacidad de software que permite a los administradores combinar discos individuales y particiones de disco y tratarlos como si fueran una sola unidad.

Periféricos

Servidores y estaciones de trabajo necesitan una combinación de CPU, la memoria del sistema, y el almacenamiento para operar. Pero estos componentes fundamentales no interactúan directamente con el mundo externo. Los periféricos son los dispositivos que proporcionan a los sistemas entradas, salidas y acceso al exterior.

La mayoría de las placas base tienen conectores externos incorporados y soporte de firmware para interfaces periféricas comunes como teclado, mouse, sonido, video y red. Las placas base recientes suelen tener un conector Ethernet para admitir redes, un conector HDMI que admite necesidades gráficas básicas y uno o más conectores USB (*Universal Serial Bus*) para casi todo lo demás. Existen varias versiones de USB con diferentes velocidades y características físicas. Varias versiones de puertos USB son comunes en una sola placa base.

Las placas base también pueden tener una o más ranuras de expansión. Las ranuras de expansión permiten a los usuarios agregar placas de circuito especiales conocidas como tarjetas de expansión. Los gráficos, el sonido y las interfaces de red son tarjetas de expansión comunes. Las tarjetas de expansión también admiten RAID e interfaces heredadas de formato especial que incluyen conexiones en serie y paralelas.

Las configuraciones *System on a Chip* (SoC) logran ventajas de potencia, rendimiento, espacio y confiabilidad sobre las configuraciones de la placa base al integrar procesadores, memoria del sistema, SSD y hardware para controlar periféricos como un solo paquete de circuito integrado. Los periféricos admitidos por las configuraciones de SoC están limitados por los componentes integrados. Por lo tanto, las configuraciones de SoC tienden a desarrollarse para usos específicos. Los teléfonos, tabletas y otros dispositivos portátiles a menudo se basan en la tecnología SoC.

Algunos sistemas incorporan periféricos. Las computadoras portátiles son similares a las estaciones

de trabajo pero incorporan periféricos de pantalla, teclado y mouse predeterminados. Los sistemas todo en uno son similares a las computadoras portátiles, pero requieren periféricos de mouse y teclado. La placa base o los controladores basados en SoC a menudo se empaquetan con periféricos integrales apropiados para un uso específico.

Controladores y archivos de dispositivo

Hasta ahora, esta lección ha presentado información sobre procesadores, memoria, discos, particionamiento, formateo y periféricos. Pero exigir a los usuarios generales que se ocupen de los detalles específicos de cada uno de los dispositivos en su sistema haría que esos sistemas no se puedan utilizar. Del mismo modo, los desarrolladores de software tendrían que modificar su código para cada dispositivo nuevo o modificado que necesiten soportar.

La solución a este problema de "tratar con los detalles" la proporciona un controlador de dispositivo. Los controladores de dispositivo aceptan un conjunto estándar de solicitudes y luego traducen esas solicitudes en las actividades de control apropiadas del dispositivo. Los controladores de dispositivo son los que le permiten a usted y a las aplicaciones leer el archivo `/home/carol/stuff` sin preocuparse de si ese archivo está en un disco duro, unidad de estado sólido, tarjeta de memoria, almacenamiento encriptado o algún otro dispositivo.

Los archivos de dispositivo se encuentran en el directorio `/dev` e identifican dispositivos físicos, acceso a dispositivos y controladores compatibles. Por convención, en los sistemas modernos que utilizan dispositivos de almacenamiento basados en SCSI o SATA, el nombre del archivo de especificación comienza con el prefijo `sd`. El prefijo va seguido de una letra como `a` o `b` que indica un dispositivo físico. Después del prefijo y el identificador del dispositivo, aparece un número que indica una partición dentro del dispositivo físico. Entonces `/dev/sda` haría referencia a todo el primer dispositivo de almacenamiento, mientras que `/dev/sda3` hará referencia a la partición 3 en el primer dispositivo de almacenamiento. El archivo de dispositivo para cada tipo de dispositivo tiene una convención de nomenclatura apropiada para el dispositivo. Si bien cubrir todas las convenciones de nomenclatura posibles está más allá del alcance de esta lección, pero es importante recordar que estas convenciones son críticas para hacer posible la administración del sistema.

Si bien está más allá del alcance de esta lección cubrir los contenidos del directorio `/dev`, es informativo mirar la entrada de un dispositivo de almacenamiento. Los archivos de dispositivo para tarjetas SD suelen utilizar `mmcblk` como prefijo:

```
$ ls -l mmcblk*
brw-rw---- 1 root disk 179, 0 Jun 30 01:17 mmcblk0
brw-rw---- 1 root disk 179, 1 Jun 30 01:17 mmcblk0p1
brw-rw---- 1 root disk 179, 2 Jun 30 01:17 mmcblk0p2
```

Los detalles de listado para un archivo de dispositivo son diferentes de los detalles de archivo típicos:

- A diferencia de un archivo o directorio, la primera letra del campo de permisos es `b`. Esto indica que los bloques se leen y escriben en el dispositivo en bloques en lugar de caracteres individuales.
- El campo de tamaño son dos valores separados por una coma en lugar de un solo valor. El primer valor generalmente indica un controlador particular dentro del núcleo y el segundo valor especifica un dispositivo específico manejado por el controlador.
- El nombre de archivo utiliza un número para el dispositivo físico, por lo que la convención de nomenclatura se adapta especificando el sufijo de partición como una `p` seguida de un dígito.

NOTE

Cada dispositivo del sistema debe tener una entrada en `/dev`. Dado que el contenido del directorio `/dev` se crea en la instalación, a menudo hay entradas para cada controlador y dispositivo posible, incluso si no existe ningún dispositivo físico.

Guía de ejercicios

1. Describa estos términos:

Procesador	
CPU	
GPU	

2. Si está ejecutando principalmente aplicaciones de edición de video (una actividad computacionalmente intensiva) ¿Qué componentes y características esperaría que tengan el mayor impacto en la usabilidad del sistema ?

CPU cores	
Velocidad del CPU	
Memoria del sistema disponible	
Sistema de almacenamiento	
GPU	
Video display	
Ningúna de las anteriores	

3. ¿Cuál esperaría que fuera el nombre del archivo del dispositivo en `/dev` para la partición 3 de la tercera unidad SATA en un sistema?

`sd3p3`	
`sdcp3`	
`sdc3`	
Ningúna de las anteriores	

Ejercicios exploratorios

1. Ejecute el comando `lsblk` en su sistema. Identifique los parámetros a continuación. Si un sistema no está disponible, considere la lista `lsblk -f` para el sistema Raspberry Pi mencionado en la sección anterior "Almacenamiento":

```
$ lsblk -f
NAME         FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat   boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

   - El tipo de dispositivos y cantidad
   - La estructura de partición de cada dispositivo
   - El tipo de sistema de archivos y montaje para cada partición

Resumen

Un sistema es la suma de sus componentes. Los diferentes componentes afectan el costo, el rendimiento y la usabilidad de diferentes maneras. Si bien hay configuraciones comunes para estaciones de trabajo y servidores, no hay una mejor configuración única.

Respuestas a los ejercicios guiados

1. Describa estos términos:

   Procesador

   Un término general que se aplica a cualquier tipo de procesador. A menudo se usa incorrectamente como sinónimo de CPU.

   CPU

   Una Unidad Central de Procesamiento (Central Processing Unit). Una unidad de procesamiento que proporciona soporte para tareas computacionales de propósito general.

   GPU

   Una unidad de procesamiento gráfico (Graphical Processing Unit). Una unidad de procesamiento optimizada para apoyar actividades relacionadas con la presentación de gráficos..

2. Si está ejecutando principalmente aplicaciones de edición de video (una actividad computacionalmente intensiva) ¿Qué componentes y características esperaría que tengan el mayor impacto en la usabilidad del sistema ?

   CPU cores

   Si. Múltiples núcleos admiten la presentación concurrente y las tareas de representación requeridas por la edición de video.

   Velocidad del CPU

   Si. La representación de video requiere una cantidad significativa de actividad computacional.

   Memoria del sistema disponible

   Probable. El video sin comprimir utilizado en la edición es grande. Los sistemas de uso general a menudo vienen con 8 gigabytes de memoria. Incluso 16 o 32 gigabytes de memoria permiten que el sistema maneje más cuadros de video sin comprimir haciendo que las actividades de edición sean más eficientes.

   Almacenamiento del sistema

   Si. Los archivos de video son grandes. La sobrecarga de las unidades SSD locales admite una transferencia más eficiente. Es probable que las unidades de red más lentas sean contraproducentes.

GPU

No. GPU afecta principalmente la presentación del video renderizado.

Video display

No. El video display impacta principalmente en la presentación del video renderizado.

Ningúna de las anteriores

No. Algunos de estos factores tienen un impacto obvio en la utilidad de su sistema.

3. ¿Cuál esperaría que fuera el nombre del archivo del dispositivo en `/dev` para la partición 3 de la tercera unidad SATA en un sistema?

`sd3p3`	Incorrecto. La unidad 3 sería `sdc` no `sd3`
`sdcp3`	Incorrecto. La partición 3 sería `3` no `p3`
`sdc3`	Correcto
Ninguna de las anteriores	Incorrecto. La respuesta correcta es una de las opciones.

Respuestas a los ejercicios exploratorios

1. Ejecute el comando `lsblk` en su sistema. Identifique los parámetros a continuación. Si un sistema no está disponible, considere la lista `lsblk -f` para el sistema Raspberry Pi mencionado en la sección anterior "Almacenamiento":

```
$ lsblk -f
NAME         FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1 vfat   boot   9304-D9FD                            /boot
+-mmcblk0p2 ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

Las siguientes respuestas se basan en la lista `lsblk -f` para el sistema Raspberry Pi anterior, así que sus respuestas pueden ser diferentes:

El tipo de dispositivos y cantidad

Hay un dispositivo: `mmcblk0`. Por convención se sabe que el `mmcblk` sería una tarjeta de memoria SD.

La estructura de partición de cada dispositivo

Hay dos particiones: `mmcblk0p1` y `mmcblk0p2`.

El tipo de sistema de archivos y montaje para cada partición

La partición 1 usa el sistema de archivos `vfat`, se utiliza para iniciar el sistema y se monta como `/boot`. La partición 2 usa el sistema de archivos `ext4`, se utiliza como sistema de archivos primario y se monta como `/`.



4.3 Donde los datos se almacenan

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 4.3

WeigImportanciaht

3

Áreas de conocimiento clave

- Programas y archivos de configuración
- Procesos
- Direcciones de memoria
- Mensajes del sistema
- Registros del sistema

Lista parcial de archivos, términos y utilidades

- `ps`, `top`, `free`
- `syslog`, `dmesg`
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`



4.3 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	4 El sistema operativo Linux
Objetivo:	4.3 Donde se almacenan los datos
Lección:	1 de 2

Introducción

Para un sistema operativo, todo se considera datos. Para Linux, todo se considera un archivo: programas, archivos normales, directorios, dispositivos de bloque (discos duros, etc.), dispositivos de caracteres (consolas, etc.), procesos del núcleo, sockets, particiones, enlaces, etc. La estructura del directorio de Linux , comenzando desde *root* `/`, es una colección de archivos que contienen datos. El hecho de que todo sea un archivo es una característica poderosa de Linux, ya que permite ajustar prácticamente todos los aspectos del sistema.

En esta lección discutiremos las diferentes ubicaciones en las que se almacenan datos importantes según lo establecido por el Linux Filesystem Hierarchy Standard (FHS). Algunas de estas ubicaciones son directorios reales que almacenan datos de forma persistente en discos, mientras que otros son pseudo sistemas de archivos cargados en la memoria que nos dan acceso a datos del subsistema del núcleo, como procesos en ejecución, uso de memoria, configuración de hardware, etc. Los datos almacenados en estos directorios virtuales son utilizados por una serie de comandos que nos permiten monitorearlos y manejarlos.

Programas y su configuración

Los datos importantes en un sistema Linux son sus programas y archivos de configuración. Los primeros son archivos ejecutables que contienen los conjuntos de instrucciones que debe ejecutar el procesador de la computadora, mientras que los segundos suelen ser documentos de texto que controlan el funcionamiento de un programa. Los archivos ejecutables pueden ser archivos binarios o archivos de texto. Los archivos de texto ejecutables se denominan scripts. Los datos de configuración en Linux también se almacenan tradicionalmente en archivos de texto, aunque existen varios estilos a la forma de representar los datos de configuración.

Dónde se almacenan los archivos binarios

Al igual que cualquier otro archivo, los archivos ejecutables se encuentran en directorios que pertenecen en última instancia de `/`. Más específicamente, los programas se distribuyen en una estructura de tres niveles: el primer nivel (`/`) incluye programas que pueden ser necesarios en modo de usuario único, el segundo nivel (`/usr`) contiene la mayoría de los programas multiusuario y el tercer nivel (`/usr/local`) se utiliza para almacenar software que no es proporcionado por la distribución y que se ha compilado localmente.

Las ubicaciones típicas para los programas incluyen:

`/sbin`
: Contiene archivos binarios esenciales para la administración del sistema, como `parted` o `ip`.

`/bin`
: Contiene archivos binarios esenciales para todos los usuarios, como `ls`, `mv` o `mkdir`.

`/usr/sbin`
: Almacena binarios para la administración del sistema, como `deluser` o `groupadd`.

`/usr/bin`
: Incluye la mayoría de los archivos ejecutables, como `free`, `pstree`, `sudo` o `man` que pueden ser utilizados por todos los usuarios.

`/usr/local/sbin`
: Se utiliza para almacenar programas instalados localmente para la administración del sistema que no son gestionados por el administrador de paquetes.

`/usr/local/bin`
: Sirve para el mismo propósito que `/usr/local/sbin` pero para programas de usuario regulares.

Recientemente, algunas distribuciones comenzaron a reemplazar `/bin` y `/sbin` con enlaces simbólicos a `/usr/bin` y `/usr/sbin`

> **NOTE** El directorio `/opt` a veces se usa para almacenar aplicaciones opcionales de terceros.

Además de estos directorios, los usuarios habituales pueden tener sus propios programas en:

- `/home/$USER/bin`
- `/home/$USER/.local/bin`

> **TIP** Puede averiguar desde qué directorios están disponibles para ejecutar binarios haciendo referencia a la variable `PATH` con `echo $PATH`. Para obtener más información sobre `PATH`, revise las lecciones sobre variables y personalización de shell.

Podemos encontrar la ubicación de los programas con el comando `which`:

```
$ which git
/usr/bin/git
```

Dónde se almacenan los archivos de configuración

El directorio `/etc`

En los primeros días de Unix había una carpeta para cada tipo de datos, como `/bin` para binarios y `/boot` para los núcleos. Sin embargo, `/etc` (que significa *etcétera*) se creó como un directorio general para almacenar cualquier archivo que no perteneciera a las otras categorías. La mayoría de estos archivos eran archivos de configuración. Con el paso del tiempo, se agregaron más y más archivos de configuración, por lo que `/etc` se convirtió en la carpeta principal para los archivos de configuración de los programas. Como se dijo anteriormente, un archivo de configuración generalmente es un archivo de texto plano local (opuesto al binario) que controla la operación de un programa.

En `/etc` podemos encontrar diferentes patrones para los nombres de los archivos de configuración:

- Archivos con una extensión *ad hoc* o sin extensión, por ejemplo

  `group`
  : Base de datos de los grupos de sistema.

`hostname`
: Nombre del equipo.

`hosts`
: Lista de direcciones IP y sus traducciones de nombres de host.

`passwd`
: Base de datos del usuario del sistema: compuesta por siete campos separados por dos puntos que proporcionan información sobre el usuario.

`profile`
: Archivo de configuración de todo el sistema para Bash.

`shadow`
: Archivo encriptado para contraseñas de usuario.

- Archivos de inicialización que terminan en `rc`:

`bash.bashrc`
: Archivo `.bashrc` en todo el sistema para shells interactivos.

`nanorc`
: Ejemplo de archivo de inicialización para GNU nano (un editor de texto simple que normalmente se funciona con cualquier distribución).

- Archivos que terminan en `.conf`:

`resolv.conf`
: Archivo de configuración para el resolver que proporciona acceso al sistema de nombres de dominio de Internet (DNS).

`sysctl.conf`
: Archivo de configuración para establecer variables del sistema para el núcleo.

- Directorios con el sufijo `.d`:

Algunos programas con un archivo de configuración único (`*.conf` o de otro modo) han evolucionado para tener un directorio dedicado `*.d` que ayuda a construir configuraciones modulares y más robustas. Por ejemplo, para configurar logrotate, encontrará `logrotate.conf`, pero también los directorios `logrotate.d`.

Este enfoque es útil en aquellos casos en que diferentes aplicaciones necesitan configuraciones para el mismo servicio específico. Si, por ejemplo, un paquete de servidor web contiene una configuración de logrotate, esta configuración ahora se puede colocar en un archivo dedicado en el directorio `logrotate.d`. Este archivo puede ser actualizado por el paquete del servidor web sin interferir con la configuración de logrotate restante. Del mismo modo, los paquetes pueden agregar tareas específicas colocando archivos en el directorio `/etc/cron.d` en lugar de modificar `/etc/crontab`.

En Debian - y derivados de Debian - este enfoque se ha aplicado a la lista de fuentes confiables leídas por la herramienta de administración de paquetes `apt`: aparte del clásico `/etc/apt/sources.list`, ahora encontramos el Directorio `/etc/apt/sources.list.d`:

```
$ ls /etc/apt/sources*
/etc/apt/sources.list
/etc/apt/sources.list.d:
```

Archivos de configuración en `HOME` (Dotfiles)

A nivel de usuario, los programas almacenan sus configuraciones en archivos ocultos del directorio de inicio de cada usuario (también representado `~`). Recuerde, los archivos ocultos comienzan con un punto (`.`), de ahí su nombre: *dotfiles*.

Algunos de estos archivos de puntos son scripts de Bash que personalizan la sesión de shell del usuario y se obtienen tan pronto como el usuario inicia sesión en el sistema:

`.bash_history`
: Almacena el historial de la línea de comando.

`.bash_logout`
: Incluye comandos para ejecutar al salir de la shell de inicio de sesión.

`.bashrc`
: Script de inicialización de Bash para shells sin inicio de sesión.Script de inicialización de Bash para shells sin inicio de sesión.

`.profile`
: Script de inicialización de Bash para shells de inicio de sesión.

> **NOTE** Consulte la lección sobre "Conceptos básicos de la línea de comandos" para obtener más información sobre Bash y sus archivos de inicio.

Los archivos de configuración de otros programas específicos del usuario se obtienen cuando se inician sus respectivos programas: `.gitconfig`, `.emacs.d`, `.ssh`, etc.

El Kernel de Linux

Antes de que se pueda ejecutar cualquier proceso, el kernel debe cargarse en un área protegida de la memoria. Después de eso, el proceso con PID `1` (en la mayoría de los casos `systemd` en la actualidad) pone en marcha la cadena de procesos, es decir, un proceso inicia otro(s) y así sucesivamente. Una vez que los procesos están activos, el kernel de Linux se encarga de asignarles recursos (teclado, mouse, discos, memoria, interfaces de red, etc.).

> **NOTE** Antes de `systemd`, `/sbin/init` siempre fue el primer proceso en un sistema Linux como parte del administrador del sistema *System V Init*. De hecho, todavía se encuentra `/sbin/init` actualmente pero vinculado a `/lib/systemd/systemd`.

Dónde se almacenan los núcleos: `/boot`

El kernel reside en `/boot` junto con otros archivos relacionados con el arranque. La mayoría de estos archivos incluyen los componentes del número de versión del núcleo en sus nombres (versión del núcleo, revisión mayor, revisión menor y número de parche).

El directorio `/boot` incluye los siguientes tipos de archivos con nombres correspondientes a la versión del kernel respectiva:

`config-4.9.0-9-amd64`
: Los ajustes de configuración para el núcleo como las opciones y los módulos que se compilaron junto con el núcleo.

`initrd.img-4.9.0-9-amd64`
: Imagen de disco RAM inicial que ayuda en el proceso de inicio al cargar un sistema de archivos raíz temporal en la memoria.

`System-map-4.9.0-9-amd64`
: El archivo `System-map` (en algunos sistemas se llamará `System.map`) contiene las direcciones de las memoria de nombres (Memory Address Locations) de los símbolos del kernel. Cada vez que se reconstruya un kernel, el contenido del archivo cambiará, ya que las ubicaciones de memoria podrían ser diferentes. El kernel utiliza este archivo para buscar las ubicaciones de la memoria

para un símbolo de kernel en particular, o viceversa.

`vmlinuz-4.9.0-9-amd64`

El kernel propiamente dicho en un formato comprimido autoextraíble que ahorra espacio (de ahí la `z` en `vmlinuz`; `vm` significa memoria virtual y comenzó a usarse cuando el kernel obtuvo soporte para memoria virtual por primera vez).

`grub`

Directorio de configuración para el gestor de arranque `grub2`.

TIP

Debido a que es una característica crítica del sistema operativo, más de un kernel y sus archivos asociados se mantienen en `/boot` en caso de que el predeterminado se vuelva defectuoso y tengamos que recurrir a una versión anterior para tratar de arrancar el sistema y arreglarlo.

El directorio `/proc`

El directorio `/proc` es uno de los llamados sistemas de archivos virtuales o pseudo, ya que su contenido no se escribe en el disco, sino que se carga en la memoria. Se llena dinámicamente cada vez que la computadora se inicia y refleja constantemente el estado actual del sistema. `/proc` incluye información sobre:

- Procesos ejecutandose
- Configuración del kernel
- Hardware del sistema

Además de todos los datos relacionados con los procesos que veremos en la próxima lección, este directorio también almacena archivos con información sobre el hardware del sistema y los ajustes de configuración del kernel. Algunos de estos archivos incluyen:

`/proc/cpuinfo`

Almacena información sobre el CPU del sistema:

```
$ cat /proc/cpuinfo
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158
model name  : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping    : 10
cpu MHz     : 3696.000
cache size  : 12288 KB
(...)
```

`/proc/cmdline`

Almacena las cadenas pasadas al núcleo en el arranque:

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-
8061c0034c74 ro quiet
```

`/proc/modules`

Muestra la lista de módulos cargados en el kernel:

```
$ cat /proc/modules
nls_utf8 16384 1 - Live 0xffffffffc0644000
isofs 40960 1 - Live 0xffffffffc0635000
udf 90112 0 - Live 0xffffffffc061e000
crc_itu_t 16384 1 udf, Live 0xffffffffc04be000
fuse 98304 3 - Live 0xffffffffc0605000
vboxsf 45056 0 - Live 0xffffffffc05f9000 (O)
joydev 20480 0 - Live 0xffffffffc056e000
vboxguest 327680 5 vboxsf, Live 0xffffffffc05a8000 (O)
hid_generic 16384 0 - Live 0xffffffffc0569000
(...)
```

El directorio `/proc/sys`

Este directorio incluye ajustes de configuración del kernel por medio de archivos y clasificados en categorías por subdirectorio:


```
$ ls /proc/sys
abi  debug  dev  fs  kernel  net  user  vm
```

La mayoría de estos archivos actúan como un interruptor, por eso solo contienen uno de los dos valores posibles: `0` o `1` (“on” u “off”). Por ejemplo:

`/proc/sys/net/ipv4/ip_forward`
El valor que habilita o deshabilita nuestra máquina para que actúe como un enrutador (reenviar paquetes):

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Sin embargo, hay algunas excepciones:

`/proc/sys/kernel/pid_max`
La cantidad maxima de PID permitada:

```
$ cat /proc/sys/kernel/pid_max
32768
```

WARNING Tenga mucho cuidado al cambiar la configuración del núcleo, ya que un valor incorrecto puede provocar un sistema inestable.

Dispositivos de hardware

Recuerde, en Linux “todo es un archivo”. Esto implica que la información del dispositivo de hardware, así como los ajustes de configuración propios del núcleo se almacenen en archivos especiales que residen en directorios virtuales.

El directorio `/dev`

El directorio *device* `/dev` contiene archivos de dispositivo o nodos para todos los dispositivos de hardware conectados. Estos archivos se utilizan como una interfaz entre los dispositivos y los procesos que los utilizan. Cada archivo de dispositivo se divide en una de dos categorías:

Dispositivos de bloque (Block devices)

Son aquellos en los que los datos se leen y escriben en bloques que pueden abordarse individualmente. Los ejemplos incluyen discos duros (y sus particiones, como `/dev/sda1`), unidades flash USB, CD, DVD, etc.

Dispositivos de carácter (Character devices)

Son aquellos en los que los datos se leen y escriben secuencialmente un carácter a la vez. Los ejemplos incluyen teclados, la consola de texto (`/dev/console`), puertos seriales (como `/dev/ttyS0`, etc.), etc.

Al enumerar los archivos del dispositivo, asegúrese de usar `ls` con el interruptor `-l` para diferenciar entre los dos. Podemos, por ejemplo, buscar discos duros y particiones:

```
# ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 may 25 17:02 /dev/sda
brw-rw---- 1 root disk 8, 1 may 25 17:02 /dev/sda1
brw-rw---- 1 root disk 8, 2 may 25 17:02 /dev/sda2
(...)
```

O para terminales en serie (TeleTYpewriter):

```
# ls -l /dev/tty*
crw-rw-rw- 1 root tty     5,  0 may 25 17:26 /dev/tty
crw--w---- 1 root tty     4,  0 may 25 17:26 /dev/tty0
crw--w---- 1 root tty     4,  1 may 25 17:26 /dev/tty1
(...)
```

Observe cómo el primer carácter es `b` para dispositivos de bloque y `c` para dispositivos de carácter.

TIP

El asterisco (*) es un carácter global que significa 0 o más caracteres. De ahí su importancia en los comandos `ls -l /dev/sd*` y `ls -l /dev/tty *` anteriores. Para obtener más información sobre estos personajes especiales, consulte la lección sobre globbing.

Además, `/dev` incluye algunos archivos especiales que son bastante útiles para diferentes propósitos de programación:

`/dev/zero`

Proporciona tantos caracteres nulos como se solicite.

`/dev/null`

Aka *bit bucket*. Descarta toda la información que se le envía.

`/dev/urandom`

Genera números pseudoaleatorios.

El directorio `/sys`

El sistema de archivos *sys* (`sysfs`) está montado en `/sys`. Se introdujo con la llegada del kernel 2.6 y significó una gran mejora en `/proc/sys`.

Los procesos deben interactuar con los dispositivos en `/dev` y por lo tanto, el núcleo necesita un directorio que contenga información sobre estos dispositivos de hardware. Este directorio es `/sys` y sus datos están ordenados en categorías. Por ejemplo, para verificar la dirección MAC de su tarjeta de red (`enp0s3`) puede usar el comando `cat` en el siguiente archivo:

```
$ cat /sys/class/net/enp0s3/address
08:00:27:02:b2:74
```

Memoria y tipos de memoria

Básicamente, para que un programa comience a ejecutarse debe cargarse en la memoria. En general, cuando hablamos de memoria nos referimos a *Random Access Memory* (RAM) y en comparación con los discos duros mecánicos tiene la ventaja de ser mucho más rápido. En el lado negativo, es volátil (es decir, una vez que la computadora se apaga, los datos desaparecen).

No obstante, cuando se trata de memoria podemos diferenciar dos tipos principales en un sistema Linux:

Memoria física

También conocido como *RAM*, son en forma de chips formados por circuitos integrados que contienen millones de transistores y condensadores. Estos a su vez, forman celdas de memoria (el componente básico de la memoria de la computadora). Cada una de estas celdas tiene un código hexadecimal asociado, una dirección de memoria, para que pueda ser referenciada cuando sea necesario.

Swap

También conocido como *swap space*, es la porción de memoria virtual que se encuentra en el disco duro y se usa cuando no hay más RAM disponible.

Por otro lado, existe el concepto de memoria virtual, que es una abstracción de la cantidad total de memoria utilizable (RAM y espacio en disco).

`free` analiza `/proc/meminfo` y muestra la cantidad de memoria libre y usada en el sistema de una manera muy clara:

```
$ free
              total        used        free      shared  buff/cache   available
Mem:        4050960     1474960     1482260       96900     1093740     2246372
Swap:       4192252           0     4192252
```

Expliquemos las diferentes columnas:

`total`
: Cantidad total de memoria física y de intercambio instalada.

`used`
: Cantidad de memoria física e intercambio actualmente en uso.

`free`
: Cantidad de memoria física e intercambio que actualmente no esta en uso.

`shared`
: Cantidad de memoria física utilizada principalmente por `tmpfs`.

`buff/cache`
: Cantidad de memoria física actualmente en uso por las memorias intermedias del núcleo y la caché.

`available`
: Estimación de cuánta memoria física está disponible para nuevos procesos.

Por defecto, `free` muestra valores en kibibytes, pero permite que una variedad de interruptores muestren sus resultados en diferentes unidades de medida. Algunas de estas opciones incluyen:

`-b`
: Bytes.

`-m`
: Mebibytes.

`-g`
: Gibibytes.

`-h`
: Formato legible por humanos.

`-h` siempre es cómodo de leer:

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3,9G        1,4G        1,5G         75M        1,0G        2,2G
Swap:          4,0G          0B        4,0G
```

NOTE Un kibibyte (KiB) equivale a 1,024 bytes mientras que un kilobytes (KB) equivale a 1000 bytes. Lo mismo es respectivamente cierto para mebibytes, gibibytes, etc.

Guía de ejercicios

1. Use el comando `which` para averiguar la ubicación de los siguientes programas y completar la tabla:

Programa	comando `which`	Ruta ejecutable (salida)	¿El usuario necesita privilegios de root?
`swapon`			
`kill`			
`cut`			
`usermod`			
`cron`			
`ps`			

2. ¿Dónde se encuentran los siguientes archivos?

Archivo	`/etc`	`~`
`.bashrc`		
`bash.bashrc`		
`passwd`		
`.profile`		
`resolv.conf`		
`sysctl.conf`		

3. Explique el significado de los elementos numéricos para el archivo del núcleo `vmlinuz-4.15.0-50-generic` que se encuentra en `/boot`:

Número de elemento	Significado
`4`	
`15`	
`0`	
`50`	

4. ¿Qué comando usarías para enumerar todos los discos duros y particiones en `/dev`?

Ejercicios exploratorios

1. Los archivos de dispositivo para discos duros se representan en función de los controladores que utilizan: `/dev/sd *` para unidades que usan SCSI (interfaz de sistema de computadora pequeña) y SATA (accesorio de tecnología avanzada en serie).
   - ¿Cómo se representaron las unidades antiguas IDE (Integrated Drive Electronics)?
   - ¿Y las modernas unidades NVMe (Non-Volatile Memory Express)?
2. Echa un vistazo al archivo `/proc/meminfo`. Compare el contenido de este archivo con la salida del comando `free` e identifique qué tecla de `/proc/meminfo` corresponde a los siguientes campos en la salida de `free`:

Salida de `free`	**campo** `/proc/meminfo`
`total`	
`free`	
`shared`	
`buff/cache`	
`available`	

Resumen

En esta lección, ha aprendido sobre la ubicación de los programas y sus archivos de configuración en un sistema Linux. Los hechos importantes para recordar son:

- Básicamente, los programas se encuentran en una estructura de directorio de tres niveles: `/`, `/usr` y `/usr/local`. Cada uno de estos niveles puede contener directorios `bin` y `sbin`.
- Los archivos de configuración se almacenan en `/etc` y `~`.
- Los archivos ocultos son los que comienzan con un punto (`.`).

También hemos discutido el kernel de Linux. Los hechos importantes son:

- Para Linux, todo es un archivo.
- El kernel de Linux se encuentra en `/boot` junto con otros archivos relacionados con el arranque.
- Para que los procesos comiencen a ejecutarse, el kernel primero debe cargarse en un área protegida de la memoria.
- El trabajo del kernel es el de asignar recursos del sistema a los procesos.
- El sistema de archivos virtual (o pseudo) `/proc` almacena datos importantes del kernel y del sistema de forma volátil.

Del mismo modo, hemos explorado dispositivos de hardware y aprendido lo siguiente:

- El directorio `/dev` almacena archivos especiales (también conocidos como nodos) para todos los dispositivos de hardware conectados: *block devices* o *character devices*. Los primeros transfieren datos en bloques y el otro un carácter a la vez.
- El directorio `/dev` también contiene otros archivos especiales como `/dev/zero`, `/dev/null` o `/dev/urandom`.
- El directorio `/sys` almacena información sobre dispositivos de hardware organizados en categorías.

Finalmente tocamos la memoria. Aprendimos:

- Un programa se ejecuta cuando se carga en la memoria.
- ¿Qué es RAM (Random Access Memory)?
- Qué es Swap?
- ¿Cómo mostrar el uso de la memoria?

Comandos usados en esta lección:

`cat`
: Concatenar o imprimir el contenido de un archivo.

`free`
: Muestra la cantidad de memoria libre y usada en el sistema.

`ls`
: Lista de contenidos del directorio.

`which`
: Mostrar ubicación del programa.

Respuestas a los ejercicios guiados

1. Use el comando `which` para averiguar la ubicación de los siguientes programas y completar la tabla:

Programa	comando `which`	Ruta ejecutable (salida)	¿El usuario necesita privilegios de root?
`swapon`	`which swapon`	`/sbin/swapon`	Si
`kill`	`which kill`	`/bin/kill`	No
`cut`	`which cut`	`/usr/bin/cut`	No
`usermod`	`which usermod`	`/usr/sbin/usermod`	Si
`cron`	`which cron`	`/usr/sbin/cron`	Si
`ps`	`which ps`	`/bin/ps`	No

2. ¿Dónde se encuentran los siguientes archivos?

File	`/etc`	`~`
`.bashrc`	No	Si
`bash.bashrc`	Si	No
`passwd`	Si	No
`.profile`	No	Si
`resolv.conf`	Si	No
`sysctl.conf`	Si	No

3. Explique el significado de los elementos numéricos para el archivo del núcleo `vmlinuz-4.15.0-50-generic` que se encuentra en `/boot`:

Número de elemento	Significado
`4`	Kernel version
`15`	Major revision
`0`	Minor revision
`50`	Patch number

4. ¿Qué comando usarías para enumerar todos los discos duros y particiones en `/dev`?


```
ls /dev/sd*
```

Respuestas a los ejercicios exploratorios

1. Los archivos de dispositivo para discos duros se representan en función de los controladores que utilizan: `/dev/sd *` para unidades que usan SCSI (interfaz de sistema de computadora pequeña) y SATA (accesorio de tecnología avanzada en serie).

   - ¿Cómo se representaron las unidades antiguas IDE (Integrated Drive Electronics)?

     `/dev/hd*`

   - ¿Y las modernas unidades NVMe (Non-Volatile Memory Express)?

     `/dev/nvme*`

2. Echa un vistazo al archivo `/proc/meminfo`. Compare el contenido de este archivo con la salida del comando `free` e identifique qué tecla de `/proc/meminfo` corresponde a los siguientes campos en la salida de `free`:

Salida de `free`	**campo** `/proc/meminfo`
`total`	`MemTotal` / `SwapTotal`
`free`	`MemFree` / `SwapFree`
`shared`	`Shmem`
`buff/cache`	`Buffers`, `Cached` and `SReclaimable`
`available`	`MemAvailable`



4.3 Lección 2

Certificación:	Linux Essentials
Versión:	1.6
Tema:	4 El sistema operativo Linux
Objetivo:	4.3 Donde se almacenan los datos
Lección:	2 de 2

Introducción

Después de explorar los programas y sus archivos de configuración, en esta lección aprenderemos cómo se ejecutan los comandos como procesos. Del mismo modo comentaremos sobre la mensajería del sistema, el uso del "kernel ring buffer" y cómo la llegada de `systemd` y su daemon — `journald` — cambió las formas en que se habían hecho las cosas con respecto al registro del sistema.

Procesos

Cada vez que un usuario emite un comando, se ejecuta un programa y se generan uno o más procesos.

Los procesos existen en una jerarquía. Después de que el kernel se carga en la memoria durante el arranque, se inicia el primer proceso que a su vez, inicia otros procesos también pueden iniciar otros procesos. Cada proceso tiene un identificador único (`PID`) y un identificador de proceso padre (`PPID`). Estos son números enteros positivos que se asignan en orden secuencial.

Explorando procesos dinámicamente: `top`

Puede obtener una lista dinámica de todos los procesos en ejecución con el comando `top`:

```
$ top

top - 11:10:29 up  2:21,  1 user,  load average: 0,11, 0,20, 0,14
Tasks:  73 total,   1 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,0 us,  0,3 sy,  0,0 ni, 99,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  1020332 total,   909492 free,    38796 used,    72044 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   873264 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   436 carol     20   0   42696   3624   3060 R  0,7  0,4   0:00.30 top
     4 root      20   0       0      0      0 S  0,3  0,0   0:00.12 kworker/0:0
   399 root      20   0   95204   6748   5780 S  0,3  0,7   0:00.22 sshd
     1 root      20   0   56872   6596   5208 S  0,0  0,6   0:01.29 systemd
     2 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kthreadd
     3 root      20   0       0      0      0 S  0,0  0,0   0:00.02 ksoftirqd/0
     5 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 kworker/0:0H
     6 root      20   0       0      0      0 S  0,0  0,0   0:00.00 kworker/u2:0
     7 root      20   0       0      0      0 S  0,0  0,0   0:00.08 rcu_sched
     8 root      20   0       0      0      0 S  0,0  0,0   0:00.00 rcu_bh
     9 root      rt   0       0      0      0 S  0,0  0,0   0:00.00 migration/0
    10 root       0 -20       0      0      0 S  0,0  0,0   0:00.00 lru-add-drain
    (...)
```

Como vimos anteriormente `top` también puede brindarnos información sobre el consumo de memoria y CPU del sistema, así como para cada proceso.

`top` le permite al usuario cierta interacción.

Por defecto, la salida se ordena por el porcentaje de tiempo de CPU utilizado por cada proceso en orden descendente. Este comportamiento puede modificarse presionando las siguientes teclas desde `top`:

`M`

Ordenar por uso de memoria.

`N`

Ordenar por número de identificación del proceso.

T

Ordenar por tiempo de ejecución.

P

Ordenar por porcentaje de uso de CPU.

Para cambiar entre orden descendente/ascendente simplemente presione `R`.

> **TIP** Una versión más elegante y fácil de usar en lugar de `top` es `htop`. Otra alternativa, quizás más exhaustiva es `atop`. Si aún no están instalados en su sistema, se recomienda que use su administrador de paquetes para instalarlos y probarlos.

Un Snapshot de los procesos: `ps`

Otro comando muy útil para obtener información sobre los procesos es `ps`. Mientras que `top` proporciona información dinámica, el comando `ps` es estática.

Si se invoca sin opciones la salida de `ps` es bastante discreta y se relaciona solo con los procesos adjuntos al shell actual:

```
$ ps
  PID TTY          TIME CMD
 2318 pts/0    00:00:00 bash
 2443 pts/0    00:00:00 ps
```

La información que se muestra tiene que ver con el identificador de proceso (`PID`), el terminal en el que se ejecuta el proceso (`TTY`), el tiempo de CPU tomado por el proceso (`TIME`) y el comando que inició el proceso(`CMD`).

Un modificador útil para `ps` es `-f` que muestra la lista de formato completo:

```
$ ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
carol     2318  1682  0 08:38 pts/1    00:00:00 bash
carol     2443  2318  0 08:46 pts/1    00:00:00 ps -f
```

En combinación con otros modificadores, `-f` muestra la relación entre los procesos padre e hijo:

```
$ ps -uf
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
carol      2318  0.0  0.1  21336  5140 pts/1    Ss   08:38   0:00 bash
carol      2492  0.0  0.0  38304  3332 pts/1    R+   08:51   0:00  \_ ps -uf
carol      1780  0.0  0.1  21440  5412 pts/0    Ss   08:28   0:00 bash
carol      2291  0.0  0.7 305352 28736 pts/0    Sl+  08:35   0:00  \_ emacs
index.en.adoc -nw
(...)
```

Del mismo modo `ps` puede mostrar el porcentaje de memoria utilizada cuando se invoca con el modificador `-v`:

```
$ ps -v
  PID TTY      STAT   TIME  MAJFL   TRS   DRS   RSS %MEM COMMAND
 1163 tty2     Ssl+   0:00      1    67 201224 5576  0.1 /usr/lib/gdm3/gdm-x-
session (...)
 (...)
```

NOTE Otro comando visualmente atractivo que muestra la jerarquía de procesos es `pstree`. Este ya esta incluído con todas las distribuciones principales.

Información del proceso en el directorio `/proc`

Ya hemos visto el sistema de archivos `/proc`. Este incluye un subdirectorio numerado para cada proceso en ejecución en el sistema (el número es el `PID` del proceso):

```
carol@debian:~# ls /proc
1    108  13   17   21   27   354  41   665  8    9
10   109  14   173  22   28   355  42   7    804  915
103  11   140  18   23   29   356  428  749  810  918
104  111  148  181  24   3    367  432  75   811
105  112  149  19   244  349  370  433  768  83
106  115  15   195  25   350  371  5    797  838
107  12   16   2    26   353  404  507  798  899
(...)
```

Por lo tanto, toda la información sobre un proceso particular se incluye dentro de su directorio. Hagamos una lista de los contenidos del primer proceso, aquel cuyo `PID` es `1` (la salida se ha truncado para facilitar la lectura):

```
# ls /proc/1/
attr         cmdline          environ  io        mem        ns
autogroup    comm             exe      limits    mountinfo  numa_maps
auxv         coredump_filter  fd       loginuid  mounts     oom_adj
...
```

Puede verificar — por ejemplo — el ejecutable del proceso:

```
# cat /proc/1/cmdline; echo
/sbin/init
```

Como puede observar, el binario que inició la jerarquía de procesos fue `/sbin/init`.

> **NOTE** Los comandos pueden ser concatenados con el punto y coma (`;`). El objetivo del comando `echo` después del "punto y coma" es proporcionar una nueva línea. Intenta ejecutar simplemente `cat /proc/1/cmdline` para ver la diferencia.

Carga del sistema

Cada proceso puede potencialmente consumir recursos del sistema. La llamada carga del sistema intenta agregar la carga general del sistema en un solo indicador numérico. Puede ver la carga actual con el comando `uptime`:

```
$ uptime
 22:12:54 up 13 days, 20:26,  1 user,  load average: 2.91, 1.59, 0.39
```

Los tres últimos dígitos indican el promedio de carga del sistema para el último minuto (`2.91`), los últimos cinco minutos (`1.59`) y los últimos quince minutos (`0.39`).

Cada uno de estos números indica cuántos procesos estaban esperando los recursos de la CPU o las operaciones de entrada/salida para completar. Esto significa que estos procesos estaban listos para ejecutarse si hubieran recibido los recursos respectivos.

Registro del sistema y mensajería del sistema

Tan pronto como el núcleo y los procesos comienzan a ejecutarse y comunicarse entre sí, se produce una gran cantidad de información. La mayor parte se envía a archivos: los llamados archivos de registro o simplemente *logs*.

Sin iniciar sesión, la búsqueda de un evento que sucedió en un servidor daría mucho dolor de cabeza a los administradores de sistemas, de ahí la importancia de tener una forma estandarizada y centralizada de realizar un seguimiento de los eventos del sistema. Además, los registros son determinantes y reveladores cuando se trata de resolución de problemas y seguridad, así como fuentes de datos confiables para comprender las estadísticas del sistema y hacer predicciones de tendencias.

Iniciar sesión con el demonio syslog

Tradicionalmente, los mensajes del sistema han sido gestionados por la instalación del registro estándar — syslog — o cualquiera de sus derivados — syslog-ng o rsyslog. El daemon de registro recopila mensajes de otros servicios o programas y los almacena en archivos de registro, generalmente en `/var/log`. Sin embargo, algunos servicios se encargan de sus propios registros (por ejemplo, el servidor web Apache HTTPD). Del mismo modo, el kernel de Linux utiliza el "ring buffer" en memoria para almacenar sus mensajes de registro.

Archivos de registro en `/var/log`

Debido a que los registros son datos que varían con el tiempo, normalmente se encuentran en `/var/log`.

Si explora `/var/log` se dará cuenta de que los nombres de los registros son, hasta cierto punto, bastante explicativos. Algunos ejemplos incluyen:

`/var/log/auth.log`
: Almacena información sobre la autenticación.

`/var/log/kern.log`
: Almacena información del kernel.

`/var/log/syslog`
: Almacena información del sistema.

`/var/log/messages`
: Almacena datos del sistema y de algunas aplicaciones.

> **NOTE** El nombre exacto y el contenido de los archivos de registro pueden variar según las distribuciones de Linux.

Accediendo a archivos de registro

Cuando explore archivos de registro, recuerde ser root (si no tiene permisos de lectura) y use un localizador como `less`;

```
# less /var/log/messages
Jun  4 18:22:48 debian liblogging-stdlog:  [origin software="rsyslogd"
swVersion="8.24.0" x-pid="285" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Jun 29 16:57:10 debian kernel: [    0.000000] Linux version 4.9.0-8-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1
SMP Debian 4.9.130-2 (2018-10-27)
Jun 29 16:57:10 debian kernel: [    0.000000] Command line:
BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64 root=/dev/sda1 ro quiet
```

Alternativamente, puede usar `tail` con el modificador `-f` para leer los mensajes más recientes del archivo y mostrar dinámicamente nuevas líneas a medida que se agregan:

```
# tail -f /var/log/messages
Jul  9 18:39:37 debian kernel: [    2.350572] RAPL PMU: hw unit of domain psys 2^-0
Joules
Jul  9 18:39:37 debian kernel: [    2.512802] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input7
Jul  9 18:39:37 debian kernel: [    2.513861] Adding 1046524k swap on /dev/sda5.
Priority:-1 extents:1 across:1046524k FS
Jul  9 18:39:37 debian kernel: [    2.519301] hid-generic 0003:80EE:0021.0001:
input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-
1/input0
Jul  9 18:39:37 debian kernel: [    2.623947] snd_intel8x0 0000:00:05.0: white list
rate for 1028:0177 is 48000
Jul  9 18:39:37 debian kernel: [    2.914805] IPv6: ADDRCONF(NETDEV_UP): enp0s3:
link is not ready
Jul  9 18:39:39 debian kernel: [    4.937283] e1000: enp0s3 NIC Link is Up 1000
Mbps Full Duplex, Flow Control: RX
Jul  9 18:39:39 debian kernel: [    4.938493] IPv6: ADDRCONF(NETDEV_CHANGE):
enp0s3: link becomes ready
Jul  9 18:39:40 debian kernel: [    5.315603] random: crng init done
Jul  9 18:39:40 debian kernel: [    5.315608] random: 7 urandom warning(s) missed
due to ratelimiting
```

Encontrará la salida en el siguiente formato:

- Timestamp

- Nombre de host del que proviene el mensaje
- Nombre del programa/servicio que generó el mensaje
- El PID del programa que generó el mensaje
- Descripción de la acción

La mayoría de los archivos de registro están escritos en texto plano; sin embargo, algunos pueden contener datos binarios, como es el caso de `/var/log/wtmp` que almacena datos relevantes para inicios de sesión exitosos. Puede usar el comando `file` para determinar cuál es el caso:

```
$ file /var/log/wtmp
/var/log/wtmp: dBase III DBT, version number 0, next free block index 8
```

Estos archivos se leen normalmente mediante comandos especiales. El comando `last` se usa para interpretar los datos en `/var/log/wtmp`:

```
$ last
carol    tty2         :0               Thu May 30 10:53   still logged in
reboot   system boot  4.9.0-9-amd64    Thu May 30 10:52   still running
carol    tty2         :0               Thu May 30 10:47 - crash  (00:05)
reboot   system boot  4.9.0-9-amd64    Thu May 30 09:11   still running
carol    tty2         :0               Tue May 28 08:28 - 14:11  (05:42)
reboot   system boot  4.9.0-9-amd64    Tue May 28 08:27 - 14:11  (05:43)
carol    tty2         :0               Mon May 27 19:40 - 19:52  (00:11)
reboot   system boot  4.9.0-9-amd64    Mon May 27 19:38 - 19:52  (00:13)
carol    tty2         :0               Mon May 27 19:35 - down   (00:03)
reboot   system boot  4.9.0-9-amd64    Mon May 27 19:34 - 19:38  (00:04)
```

NOTE Similar a `/var/log/wtmp`, `/var/log/btmp` almacena información sobre intentos fallidos de inicio de sesión y el comando especial para leer su contenido es `lastb`.

Rotación de registro

Los archivos de registro pueden crecer mucho en unas pocas semanas o meses y ocupar todo el espacio libre en disco. Para abordar esto, se utiliza la utilidad `logrotate`. Este realiza la rotación de registros o ciclos que implica acciones tales como mover archivos de registro a un nuevo nombre, archivarlos y/o comprimirlos, a veces enviándolos por correo electrónico al administrador del sistema y eventualmente eliminándolos a medida que avanzan. Las convenciones utilizadas para nombrar estos archivos de registro son diversas (por ejemplo, agregar un sufijo con la fecha); sin

embargo es común observar un sufijo con un número entero:

```
# ls /var/log/apache2/
access.log  error.log  error.log.1  error.log.2.gz  other_vhosts_access.log
```

Tenga en cuenta que `error.log.2.gz` ya se ha comprimido con gunzip (de ahí el sufijo` .gz`).

Kernel Ring Buffer

El kernel ring buffer es una estructura de datos de tamaño fijo que registra los mensajes de arranque del núcleo, así como cualquier mensaje en vivo del kernel. La función de este buffer — uno muy importante — es el de registrar todos los mensajes del kernel producidos en el arranque — cuando el `syslog` no está todavía disponible. El comando `dmesg` imprime el ring buffer del kernel (que solía estar también almacenado en `/var/log/dmesg`). Debido a la extensión del ring buffer, este comando se usa normalmente en combinación con la utilidad de filtrado de texto `grep` o un paginador como `less`. Por ejemplo, para buscar mensajes de arranque:

```
$ dmesg | grep boot
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[    0.000000] smpboot: Allowing 1 CPUs, 0 hotplug CPUs
[    0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[    0.144986] AppArmor: AppArmor disabled by boot time parameter
(...)
```

NOTE A medida que el kernel ring buffer crece con nuevos mensajes, con el tiempo los más antiguos se eliminiran.

El diario del sistema: systemd-journald

A partir de 2015, systemd reemplazó a SysV Init como administrador de servicio y sistema de facto en la mayoría de las principales distribuciones de Linux. Como consecuencia, el daemon journal — journald — se ha convertido en el componente de registro estándar reemplazando a syslog en la mayoría de los aspectos. Los datos ya no se almacenan en texto plano sino en forma binaria. Por lo tanto, la utilidad `journalctl` es necesaria para leer los registros. Además de eso, journald es compatible con syslog y puede integrarse con este.

`journalctl` es la utilidad que se utiliza para leer y consultar la base de datos del journal (diario) de Systemd. Si es invocada sin opciones, imprime el journal completo:

```
# journalctl
-- Logs begin at Tue 2019-06-04 17:49:40 CEST, end at Tue 2019-06-04 18:13:10 CEST.
--
jun 04 17:49:40 debian systemd-journald[339]: Runtime journal (/run/log/journal/)
is 8.0M, max 159.6M, 151.6M free.
jun 04 17:49:40 debian kernel: microcode: microcode updated early to revision 0xcc,
date = 2019-04-01
Jun 04 17:49:40 debian kernel: Linux version 4.9.0-8-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
Jun 04 17:49:40 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64
root=/dev/sda1 ro quiet
(...)
```

Sin embargo, si se invoca con los modificadores `-k` o `--dmesg`, será equivalente a usar el comando `dmesg`:

```
# journalctl -k
[    0.000000] Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc
version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP Debian 4.9.168-1+deb9u2
(2019-05-13)
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
(...)
```

Otras opciones interesantes para `journalctl` incluyen:

`-b`, `--boot`
: Muestra información de arranque.

`-u`
: Muestra mensajes sobre una unidad especificada. Aproximadamente una unidad se puede definir como cualquier recurso manejado por systemd. Por ejemplo, `journalctl -u apache2.service` se usa para leer mensajes sobre el servidor web `apache2`.

`-f`
: Muestra los mensajes de diario más recientes y sigue imprimiendo nuevas entradas a medida que se agregan al diario, de forma muy similar a `tail -f`.

Ejercicios guiados

1. Eche un vistazo a la siguiente lista de `top` y responda las siguientes preguntas:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1020332 total,   698700 free,   170664 used,   150968 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   710956 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424  34256 S  6.3 13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132   2572 S  0.3  0.4   0:00.44 apache2
  734 root      20   0   95212   7004   6036 S  0.3  0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R  0.3  0.4   0:00.03 top
    1 root      20   0   56988   6688   5240 S  0.0  0.7   0:00.42 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.09 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.87 kworker/0:0
(...)
```

   - ¿Qué procesos ha iniciado el usuario `carol`?

   - ¿Qué directorio virtual de `/proc` debe visitar para buscar datos sobre el comando `top`?

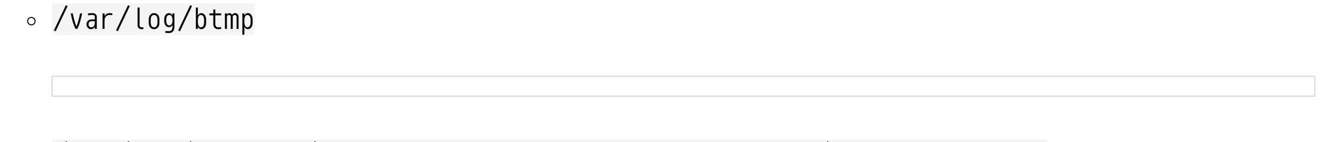
   - ¿Qué proceso se ejecutó primero? ¿Cómo puede saber ?

   - Complete la tabla especificando en qué área de salida `top` se encuentra la siguiente información:

Información sobre ...	Área de resumen	Área de tareas
Memoria		
Swap		
PID		

Información sobre …	Área de resumen	Área de tareas
CPU time		
Comandos		

2. ¿Qué comando se usa para leer los siguientes registros binarios?
   - `/var/log/wtmp`



   - `/var/log/btmp`

   - `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

3. En combinación con `grep`, ¿qué comandos usaría para averiguar la siguiente información sobre su sistema Linux?
   - ¿Cuándo se reinició el sistema por última vez (`wtmp`) ?

   - ¿Qué discos duros están instalados (`kern.log`) ?

   - ¿Cuando ocurrió el último inicio de sesión (`auth.log`) ?

4. ¿Cuales dos comandos usarías para mostrar el kernel ring buffer?

5. Indique a dónde pertenecen los siguientes mensajes de registro:
   - `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting…`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

6. ¿Tiene información para consultar el comando `journalctl` sobre las siguientes unidades?

Unidad	Comando
`ssh`	
`networking`	
`rsyslog`	
`cron`	

Ejercicios exploratorios

1. Reconsidere el resultado de `top` de los ejercicios guiados y responda las siguientes preguntas:
   - ¿Qué dos pasos seguirías para matar el servicio web *apache*?
   - En el área de resumen, ¿cómo podría mostrar la información sobre la memoria física e swap utilizando las barras de progreso?
   - Ahora, ordena los procesos por uso de memoria:
   - Ahora que tiene información de memoria en barras de progreso y procesos ordenados por uso de memoria, guarde estas configuraciones para que las obtenga como predeterminadas la próxima vez que use `top`:
   - ¿Qué archivo almacena los ajustes de configuración de `top`? ¿Dónde se encuentra? ¿Cómo puedes verificar su existencia?
2. Aprende sobre el comando `exec` en Bash. Intente demostrar su funcionalidad iniciando una sesión Bash encontrando el proceso Bash con `ps`, luego ejecute `exec /bin/sh` y busque el proceso con el mismo PID nuevamente.
3. Siga estos pasos para explorar los eventos del kernel y la administración dinámica de dispositivos de udev:
   - Conecte en caliente una unidad USB a su computadora. Ejecute `dmesg` y preste atención a las últimas líneas. ¿Cuál es la línea más reciente?
   - Teniendo en cuenta el resultado del comando anterior, ejecute `ls /dev/sd*` y asegúrese de que su unidad USB aparezca en la lista. ¿Cuál es el resultado?
   - Ahora retire la unidad USB y ejecute `dmesg` nuevamente. ¿Cómo se lee la línea más reciente?

- Ejecute `ls /dev/sd*` nuevamente y asegúrese de que su dispositivo desapareció de la lista. ¿Cuál es el resultado?

Resumen

En el contexto del almacenamiento de datos, los siguientes temas se han discutido en esta lección: administración de procesos, registro y mensajería del sistema.

En cuanto a la gestión de procesos, hemos aprendido lo siguiente:

- Los programas generan procesos y los procesos existen en una jerarquía.
- Cada proceso tiene un identificador único (`PID`) y un identificador de proceso padre (`PPID`).
- `top` es un comando muy útil para explorar de forma dinámica e interactiva los procesos en ejecución del sistema.
- `ps` se puede usar para obtener una instantánea de los procesos actuales en ejecución en el sistema.
- El directorio `/proc` incluye directorios para cada proceso en ejecución en el sistema con el nombre de sus PID.
- El concepto de promedio de carga del sistema que es muy útil para verificar la utilización/sobrecarga de la CPU.

Con respecto al registro del sistema, debemos recordar que:

- Un registro es un archivo donde se registran los eventos del sistema. Los registros son invaluables cuando se trata de solucionar problemas.
- El registro ha sido manejado tradicionalmente por servicios especiales como syslog, syslog-ng o rsyslog. Sin embargo, algunos programas usan sus propios demonios de registro.
- Debido a que los registros son datos variables, se mantienen en `/var` y a veces sus nombres pueden darle una pista sobre su contenido (`kern.log`, `auth.log`, etc.)
- La mayoría de los registros están escritos en texto plano y se pueden leer con cualquier editor de texto siempre que tenga los permisos correctos. Sin embargo, algunos de ellos son binarios y deben leerse con comandos especiales.
- Para evitar problemas con el espacio en disco, la utilidad logrotate lleva a cabo *log rotation*.
- En cuanto al kernel, utiliza una estructura de datos - kernel ring buffer - donde se guardan los mensajes de arranque (los mensajes antiguos se desvanecen con el tiempo).
- El administrador de sistema y servicio systemd reemplazó System V init en prácticamente todas las distribuciones con journald convirtiéndose en el servicio de registro estándar.
- Para leer el diario de systemd, se necesita la utilidad `journalctl`.

Comandos usados en esta lección:

`cat`
: Concatenar o imprimir el contenido de un archivo.

`dmesg`
: Imprime el kernel ring buffer.

`echo`
: Mostrar una línea de texto o una nueva línea.

`file`
: Determinar el tipo de archivo.

`grep`
: Imprimir líneas que coinciden con un patrón.

`last`
: Mostrar una lista de los últimos usuarios registrados.

`less`
: Muestra el contenido del archivo una página a la vez.

`ls`
: Lista de contenidos del directorio.

`journalctl`
: Consulta el diario `systemd`.

`tail`
: Mostrar las últimas líneas de un archivo.

Respuestas a los ejercicios guiados

1. Eche un vistazo a la siguiente lista de `top` y responda las siguientes preguntas:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1020332 total,   698700 free,   170664 used,   150968 buff/cache
KiB Swap:  1046524 total,  1046524 free,        0 used.   710956 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424  34256 S  6.3 13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132   2572 S  0.3  0.4   0:00.44 apache2
  734 root      20   0   95212   7004   6036 S  0.3  0.7   0:00.36 sshd
  887 carol     20   0   46608   3680   3104 R  0.3  0.4   0:00.03 top
    1 root      20   0   56988   6688   5240 S  0.0  0.7   0:00.42 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.09 ksoftirqd/0
    4 root      20   0       0      0      0 S  0.0  0.0   0:00.87 kworker/0:0
(...)
```

- ¿Qué procesos ha iniciado el usuario `carol`?

  Respuesta: Sólo una: `top`.

- ¿Qué directorio virtual de `/proc` debe visitar para buscar datos sobre el comando `top`?

  Respuesta: `/proc/887`

- ¿Qué proceso se ejecutó primero? ¿Cómo puede saber ?

  Respuesta: `systemd`. Porque es el que tiene `PID` #1.

- Complete la tabla especificando en qué área de salida `top` se encuentra la siguiente información:

Información sobre ...	Área de resumen	Área de tareas
Memoria	Si	Si
Swap	Si	No

Información sobre …	Área de resumen	Área de tareas
`PID`	No	Si
CPU time	Si	Si
Comandos	No	Si

2. ¿Qué comando se usa para leer los siguientes registros binarios?
   - `/var/log/wtmp`

     Respuesta: `last`
   - `/var/log/btmp`

     Respuesta: `lastb`
   - `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

     Respuesta: `journalctl`
3. En combinación con `grep`, ¿qué comandos usaría para averiguar la siguiente información sobre su sistema Linux?
   - ¿Cuándo se reinició el sistema por última vez (`wtmp`) ?

     Respuesta: `last`
   - ¿Qué discos duros están instalados (`kern.log`) ?

     Respuesta: `less /var/log/kern.log`
   - ¿Cuando ocurrió el último inicio de sesión (`auth.log`) ?

     Respuesta: `less /var/log/auth.log`
4. ¿Cuales dos comandos usarías para mostrar el kernel ring buffer?

   `dmesg` y `journalctl -k` (also `journalctl --dmesg`).
5. Indique a dónde pertenecen los siguientes mensajes de registro:
   - `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	X
`/var/log/messages`	

- `Jul 10 11:23:58 debian kernel: [ 1.923349] usbhid: USB HID core driver`

`/var/log/auth.log`	
`/var/log/kern.log`	X
`/var/log/syslog`	
`/var/log/messages`	X

`Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)`

`/var/log/auth.log`	X
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	

- `Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting…`

`/var/log/auth.log`	
`/var/log/kern.log`	
`/var/log/syslog`	
`/var/log/messages`	X

6. ¿Tiene información para consultar el comando `journalctl` sobre las siguientes unidades?

Unidad	**Comando**
`ssh`	`journalctl -u ssh.service`
`networking`	`journalctl -u networking.service`
`rsyslog`	`journalctl -u rsyslog.service`
`cron`	`journalctl -u cron.service`

Respuestas a los ejercicios exploratorios

1. Reconsidere el resultado `top` de los ejercicios guiados y responda las siguientes preguntas:

   - ¿Cuales son los dos pasos seguientes para matar el servicio web *apache*?

     Primero, presione `k`, luego proporcione un valor a `kill`.

   - En el área de resumen, ¿cómo podría mostrar la información sobre la memoria física y swap utilizando las barras de progreso?

     Al presionar `m` una o dos veces.

   - Ahora, ordena los procesos por uso de memoria:

     `M`

   - Ahora que tiene la información de la memoria en barras de progreso y procesos ordenados por uso de memoria, guarde estas configuraciones para que las obtenga como predeterminadas la próxima vez que use `top`:

     `W`

   - ¿Qué archivo almacena los ajustes de configuración de `top`? ¿Dónde se encuentran? ¿Cómo puedes verificar su existencia?

     El archivo es `~/.config/procps/toprc` y se localiza en el directorio de inicio del usuario (~). Como es un archivo oculto (reside en un directorio cuyo nombre comienza con un punto), podemos verificar su existencia con `ls -a` (lista todos los archivos). Este archivo puede generarse presionando `Shift` + `W` mientras estas en `top`.

2. Aprende sobre el comando `exec` en Bash. Intente demostrar su funcionalidad iniciando una sesión Bash encontrando el proceso Bash con `ps`, luego ejecute `exec /bin/sh` y busque el proceso con el mismo PID nuevamente.

   `exec` reemplaza un proceso con otro comando. En el siguiente ejemplo podemos ver que el proceso Bash se reemplaza por `/bin/sh` (en lugar de que `/bin/sh` se convierta en un proceso hijo):

```
$ echo $$
19877
$ ps auxf | grep 19877 | head -1
carol  19877  0.0  0.0   7448  3984 pts/25   Ss   21:17   0:00  \_ bash
$ exec /bin/sh
sh-5.0$ ps auxf | grep 19877 | head -1
carol  19877  0.0  0.0   7448  3896 pts/25   Ss   21:17   0:00  \_ /bin/sh
```

3. Siga estos pasos para explorar los eventos del kernel y la administración dinámica de dispositivos de udev:

   - Conecte en caliente una unidad USB a su computadora. Ejecute `dmesg` y preste atención a las últimas líneas. ¿Cuál es la línea más reciente?

     Debería obtener algo similar a `[ 1967.700468] sd 6:0:0:0: [sdb] Attached SCSI removable disk`.

   - Teniendo en cuenta el resultado del comando anterior, ejecute `ls /dev/sd*` y asegúrese de que su unidad USB aparezca en la lista. ¿Cuál es el resultado?

     Dependiendo de la cantidad de dispositivos conectados a su sistema, debería obtener algo como `/dev/sda /dev/sda1 /dev/sdb /dev/sdb1 /dev/sdb2`. En nuestro caso, encontramos nuestra unidad USB (`/dev/sdb`) y sus dos particiones (`/dev/sdb1` y `/dev/sdb2`).

   - Ahora retire la unidad USB y ejecute `dmesg` nuevamente. ¿Cómo se lee la línea más reciente?

     Debería obtener algo similar a `[ 2458.881695] usb 1-9: USB disconnect, device number 6`.

   - Ejecute `ls /dev/sd *` nuevamente y asegúrese de que su dispositivo desapareció de la lista. ¿Cuál es el resultado?

     En nuestro caso, `/dev/sda /dev/sda1`.



4.4 Tu ordenador en la red

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 4.4

Importancia

2

Áreas de conocimiento clave

- Internet, red, routers
- Consultar la configuración del cliente DNS
- Consultar la configuración de red

Lista parcial de archivos, términos y utilidades

- `route`, `ip route show`
- `ifconfig`, `ip addr show`
- `netstat`, `ss`
- `/etc/resolv.conf`, `/etc/hosts`
- `IPv4`, `IPv6`
- `ping`
- `host`



4.4 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	4 El sistema operativo Linux
Objetivo:	4.4 Su computadora en la red
Lección:	1 de 1

Introducción

En el mundo actual, cualquier tipo de dispositivo informático intercambia información por medio de redes. En el corazón mismo del concepto de redes de computadoras están las conexiones físicas entre un dispositivo y sus pares. Estas conexiones se denominan *enlaces* y son la conexión más básica entre dos dispositivos diferentes. Los enlaces se pueden establecer a través de diversos medios, como cables de cobre, fibras ópticas, ondas de radio o láser.

Cada enlace está conectado por medio de una interfaz del dispositivo. Cada dispositivo puede tener múltiples interfaces y por lo tanto estar conectado a múltiples enlaces. A través de estos enlaces, las computadoras pueden formar una red; una pequeña comunidad de dispositivos que pueden conectarse directamente entre sí. Existen numerosos ejemplos de tales redes en el mundo. Para poder comunicarse más allá del alcance de una red de capa de enlace, los dispositivos usan enrutadores. Piense en las redes de capa de enlace como islas conectadas por enrutadores que conectan las islas, igual que los puentes, la información tiene que viajar para llegar a un dispositivo que forma parte de una isla remota.

Este modelo conduce a varias capas diferentes de redes:

Capa de enlace

Maneja la comunicación entre dispositivos conectados directamente.

Capa de red

Maneja el enrutamiento fuera de las redes individuales y el direccionamiento único de dispositivos más allá de una red única de capa de enlace.

Capa de aplicación

Permite que los programas se conecten entre sí.

En principio, las redes de computadoras usaban los mismos métodos de comunicación que los teléfonos, ya que tenían conmutación de circuitos. Esto significa que se debe formar un enlace dedicado y directo entre dos nodos para que puedan comunicarse. Este método funcionó bien, sin embargo, requirió todo el espacio en un enlace dado para que solo dos hosts se podían comunicar.

Finalmente, las redes de computadoras se trasladaron a algo llamado *paquetes de conmutación*. En este método, los datos se agrupan por un encabezado que contiene información acerca de dónde proviene la información y hacia dónde va. La información de contenido real está contenida en este marco y se envía a través del enlace al destinatario indicado en el encabezado del marco. Esto permite que múltiples dispositivos compartan un solo enlace y se comuniquen casi simultáneamente.

Redes de capa de enlace

El trabajo de cualquier paquete es llevar información desde la fuente a su destino a través de un enlace que conecta ambos dispositivos. Estos dispositivos necesitan una forma de identificarse entre sí. Este es el propósito de una *dirección de capa de enlace*. En una red ethernet, se usan *Media Access Control Addresses* (MAC) para identificar dispositivos individuales. Una dirección MAC consta de 48 bits. No son necesariamente únicos a nivel mundial y no pueden utilizarse para dirigirse a pares fuera del enlace actual. Por lo tanto, estas direcciones no se pueden usar para enrutar paquetes a otros enlaces. El destinatario de un paquete verifica si la dirección de destino coincide con su propia capa de enlace y, si lo hace, procesa el paquete. De lo contrario, el paquete se descarta. La excepción a esta regla son los paquetes de difusión (un paquete enviado a todos en una red local determinada) que siempre se aceptan.

El comando `ip link show` muestra una lista de todas las interfaces de red disponibles y sus direcciones de capa de enlace, así como alguna otra información sobre el tamaño máximo de paquete:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
```

El resultado anterior muestra que el dispositivo tiene dos interfaces, `lo` y `ens33`. `lo` es el *loopback device* y tiene la dirección MAC `00:00:00:00:00:00` mientras que `ens33` es una interfaz de Ethernet y tiene la dirección MAC `00:0c:29:33:3b:25`.

Redes IPv4

Para visitar sitios web como Google o Twitter, consultar correos electrónicos o permitir que las empresas se conecten entre sí; los paquetes deberán poder moverse de una red de capa de enlace a otra. A menudo, estas redes están conectadas de manera indirecta, con varias redes de capa de enlace intermedio que los paquetes tienen que cruzar para llegar a su destino real.

Las direcciones de capa de enlace de una interfaz de red no se pueden usar fuera de esa red de capa de enlace específica. Dado que esta dirección no tiene importancia para los dispositivos en otras redes de capa de enlace, se necesita una forma de direcciones globalmente únicas para implementar el enrutamiento. Este esquema de direccionamiento, junto con el concepto general de enrutamiento, se implementa mediante *Internet Protocol* (IP).

NOTE

Un *protocolo* es un conjunto de procedimientos de como hacer algo para que todas las partes que siguen el protocolo sean compatibles entre sí. Un protocolo puede verse como la definición de un estándar. En informática, el Protocolo de Internet es un estándar acordado por todos para que diferentes dispositivos producidos por diferentes fabricantes puedan comunicarse entre sí.

Direcciones IPv4

Las direcciones IP, como las direcciones MAC, son una forma de indicar de dónde viene un paquete de datos y hacia dónde va. IPv4 fue el protocolo original. Las direcciones IPv4 tienen 32 bits de ancho, lo que da un número máximo teórico de 4,294,967,296 direcciones. Sin embargo, el número de esas direcciones que pueden usar los dispositivos es mucho menor, ya que algunos rangos están reservados para casos de uso especiales, como las direcciones de difusión (que se utilizan para llegar a todos los participantes de una red específica), las direcciones de multidifusión (similares a las direcciones de difusión, pero un el dispositivo debe sintonizarse como una radio) o aquellos

reservados para uso privado.

En un formato legible para los humanos, las direcciones IPv4 se identifican en cuatro dígitos separados por un punto. Cada dígito puede variar de `0` a `255`. Por ejemplo, siguiente dirección IP:

```
192.168.0.1
```

Técnicamente, cada uno de estos dígitos representa ocho bits individuales. Por lo tanto, esta dirección también podría escribirse así:

```
11000000.10101000.00000000.00000001
```

En la práctica se usa la notación decimal como se ve en el ejemplo anterior, sin embargo, esto es una representación para comprender como subneteo.

Subredes IPv4

Para admitir el enrutamiento, las direcciones IP se pueden dividir en dos partes: la red y los host. La primera parte identifica la red en la que se encuentra el dispositivo y se utiliza para enrutar paquetes. La parte de los host se usa para identificar un dispositivo específico en una red y entregar el paquete a su destinatario específico una vez que haya seleccionado su red de capa de enlace.

Las direcciones IP se pueden dividir en subredes y en hosts en cualquier momento. La llamada máscara de subred define dónde ocurre esta división. Reconsideremos la representación binaria de la dirección IP del ejemplo anterior:

```
11000000.10101000.00000000.00000001
```

Ahora para esta dirección IP, la máscara de subred establece cada bit perteneciente a la red con valor `1` y cada bit que pertenece a los hosts con valor `0`:

```
11111111.11111111.11111111.00000000
```

En la práctica, la máscara de red se escribe en la notación decimal:

```
255.255.255.0
```

Esto significa que esta red varía de `192.168.0.0` a `192.168.0.255`. Tenga en cuenta que los primeros tres números tienen todos los bits establecidos en la máscara de red, permanecen sin cambios en las direcciones IP.

Finalmente, hay una notación alternativa para la máscara de subred, que se denomina *Classless Inter-Domain Routing* (CIDR). Esta notación solo indica cuántos bits se configuran en la máscara de subred y agrega este número a la dirección IP. En el ejemplo, 24 de 32 bits están configurados en `1` en la máscara de subred. Esto se puede expresar en notación CIDR como `192.168.0.1/24`

Direcciones IPv4 privadas

Como se mencionó anteriormente, algunas secciones del espacio de direcciones IPv4 están reservadas para casos de uso especiales. Uno de estos casos de uso son las asignaciones de direcciones privadas. Las siguientes subredes están reservadas para direccionamiento privado:

- `10.0.0.0/8`
- `172.16.0.0/12`
- `192.168.0.0/16`

Las direcciones que salen de estas subredes pueden ser utilizadas por cualquier persona. Sin embargo, estas subredes no se pueden enrutar en Internet público, ya que potencialmente son utilizadas por numerosas redes al mismo tiempo.

La mayoría de las redes usan estas direcciones internas permitiendo la comunicación interna sin la necesidad de ninguna asignación. La mayoría de las conexiones a Internet hoy en día solo vienen con una única dirección IPv4 externa. Los enrutadores asignan todas las direcciones internas a esa única dirección IP externa cuando reenvían paquetes a Internet. Esto se llama *Network Address Translation* (NAT). El caso especial de NAT en el que un enrutador asigna direcciones internas a una sola dirección IP externa a veces se llama *enmascaramiento (masquerading)*. Esto permite que cualquier dispositivo en la red interna establezca nuevas conexiones con cualquier dirección IP global en Internet.

NOTE Con el enmascaramiento, los dispositivos internos no pueden ser referenciados desde Internet ya que no tienen una dirección válida globalmente. Sin embargo, esto no es una característica de seguridad. Incluso cuando se usa el enmascaramiento se necesita un firewall.

Configuración de dirección IPv4

Hay dos formas principales de configurar las direcciones IPv4 en una computadora. Asignando

direcciones manualmente o usando el *Dynamics Host Configuration Protocol* (DHCP) para la configuración automática.

Cuando se utiliza DHCP, un servidor central controla qué direcciones se entregan a qué dispositivos. El servidor también puede suministrar a los dispositivos otra información sobre la red, como las direcciones IP de los servidores DNS, la dirección IP del enrutador predeterminado o en el caso de configuraciones más complicadas, iniciar un sistema operativo desde la red. DHCP está habilitado de manera predeterminada en muchos sistemas, por lo tanto, es probable que ya tenga una dirección IP cuando esté conectado a una red.

Las direcciones IP también se pueden agregar manualmente a una interfaz usando el comando `ip addr add`. Aquí, agregamos la dirección `192.168.0.5` a la interfaz `ens33`. La red utiliza la máscara de red `255.255.255.0` que es igual a `/24` en notación CIDR:

```
$ sudo ip addr add 192.168.0.5/255.255.255.0 dev ens33
```

Ahora podemos verificar que la dirección fue agregada usando el comando `ip addr show`:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

La salida anterior muestra tanto la interfaz `lo` como la interfaz `ens33` con su dirección asignada con el comando anterior.

Para verificar la accesibilidad de un dispositivo se puede usar el comando `ping`. Envía un tipo de mensaje especial llamado *echo request* en el que el remitente solicita una respuesta al destinatario. Si la conexión entre los dos dispositivos se puede establecer con éxito, el destinatario enviará una respuesta de echo verificando así la conexión:

```
$ ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.16 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.85 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=3.41 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
rtt min/avg/max/mdev = 1.849/2.473/3.410/0.674 ms
```

El parámetro `-c 3` hace que `ping` se detenga después de enviar tres solicitudes de echo. De lo contrario, `ping` continúa ejecutándose para siempre y debe detenerse presionando `Ctrl` + `C`.

Enrutamiento IPv4

El enrutamiento es el proceso en el que un paquete llega desde la red de origen a la red de destino. Cada dispositivo mantiene una tabla de enrutamiento que contiene información sobre a qué red se puede acceder a través de la conexión del dispositivo para vincular redes de capa y a qué red se puede acceder al pasar paquetes a un enrutador. Finalmente, una ruta predeterminada define un enrutador que recibe todos los paquetes que no coinciden con ninguna otra ruta.

Al establecer una conexión, el dispositivo busca la dirección IP del objetivo en su tabla de enrutamiento. Si una entrada coincide con la dirección, el paquete se envía a la red de capa de enlace respectiva o se pasa al enrutador indicado en la tabla de enrutamiento.

Los enrutadores también mantienen tablas de enrutamiento, al recibir un paquete un enrutador también busca la dirección de destino en su propia tabla de enrutamiento y envía el paquete al siguiente enrutador. Esto se repite hasta que el paquete llega al enrutador en la red de destino. Cada enrutador involucrado en este viaje se llama *hop*. Este último enrutador encuentra un enlace conectado directo para la dirección de destino en su tabla de enrutamiento y envía los paquetes a su interfaz de destino.

La mayoría de las redes domésticas solo tienen una salida, el enrutador singular que tuvieron del *proveedor de servicios de Internet* (ISP). En este caso, un dispositivo solo reenvía todos los paquetes que no son para la red interna directamente al enrutador doméstico que luego lo enviará al enrutador del proveedor para su posterior reenvío. Este es un ejemplo de la ruta predeterminada.

El comando `ip route show` enumera la tabla de enrutamiento IPv4 actual:


```
$ ip route show
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Para agregar una ruta predeterminada, todo lo que se necesita es la dirección interna del enrutador que será la puerta de enlace predeterminada. Si por ejemplo, el enrutador tiene la dirección `192.168.0.1`, entonces el siguiente comando lo configura como una ruta predeterminada:

```
$ sudo ip route add default via 192.168.0.1
```

Para verificar, ejecute `ip route show` nuevamente:

```
$ ip route show
default via 192.168.0.1 dev ens33
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Redes IPv6

IPv6 fue diseñado para hacer frente a las deficiencias de IPv4, principalmente la falta de direcciones a medida que se conectaban más y más dispositivos. Sin embargo, IPv6 también incluye otras características como nuevos protocolos para la configuración automática de la red. En lugar de 32 bits por dirección, IPv6 usa 128. Esto permite aproximadamente 2^{128} direcciones. Sin embargo, al igual que IPv4, la cantidad de direcciones utilizables globalmente únicas es mucho menor debido a que las secciones de la asignación se reservan para otros usos. Esto significa que hay más que suficientes direcciones públicas para cada dispositivo que se encuentra actualmente conectado a Internet y para muchos más por venir, lo que reduce la necesidad de enmascararse y sus problemas, como el retraso durante la traducción y la imposibilidad de conectarse directamente a dispositivos enmascarados.

Direcciones IPv6

Las direcciones usan 8 grupos de 4 dígitos hexadecimales cada uno separado por dos puntos:

```
2001:0db8:0000:abcd:0000:0000:0000:7334
```

> **NOTE** Los dígitos hexadecimales van de `0` a `f`, por lo que cada dígito puede contener uno de 16 valores diferentes.

Para más fácilIdad, se pueden eliminar los ceros iniciales de cada grupo cuando se escriben, sin embargo, cada grupo debe contener al menos un dígito:

```
2001:db8:0:abcd:0:0:0:7334
```

Cuando varios grupos que contienen solo ceros que se encuentran uno tras otro, pueden reemplazarse por `::`:

```
2001:db8:0:abcd::7334
```

Sin embargo, esto solo puede suceder una vez en cada dirección.

Prefijo IPv6

Los primeros 64 bits de una dirección IPv6 se conocen como *prefijo de ruta (routing prefix)*. Los enrutadores utilizan este prefijo para determinar a qué red pertenece un dispositivo y por lo tanto a qué ruta deben enviarse los datos. La división en subredes siempre ocurre dentro del prefijo. Los ISP generalmente distribuyen prefijos /48 o /58 a sus clientes dejándo 16 o 8 bits para su subred interna.

Hay tres tipos principales de prefijos en IPv6:

Global Unique Address
: En donde el prefijo se asigna desde los bloques reservados para direcciones globales. Estas direcciones son válidas en todo internet.

Unique Local Address
: No se puede enrutar en internet. Sin embargo, pueden enrutarse internamente dentro de una organización. Estas direcciones se usan dentro de una red para garantizar que los dispositivos aún tengan una dirección incluso cuando no haya conexión a Internet. Son el equivalente de los rangos de direcciones privadas de IPv4. Los primeros 8 bits son siempre `fc` o `fd` seguidos de 40 bits generados aleatoriamente.

Link Local Address
: Solo son válidos en un enlace en particular. Cada interfaz de red compatible con IPv6 tiene una de esas direcciones, comenzando con `fe80`. IPv6 utiliza estas direcciones internamente para solicitar direcciones adicionales mediante la configuración automática y para buscar otras

computadoras en la red mediante el protocolo Neighbor Discovery (ND).

Identificador de interfaz IPv6

Si bien el prefijo determina en qué red reside un dispositivo, el identificador de interfaz se utiliza para enumerar los dispositivos dentro de una red. Los últimos 64 bits en una dirección IPv6 forman el identificador de interfaz, al igual que los últimos bits de una dirección IPv4.

Cuando una dirección IPv6 se asigna manualmente, el identificador de interfaz se configura como parte de la dirección. Cuando se usa la configuración de dirección automática, el identificador de la interfaz se elige al azar o se deriva de la dirección de la capa de enlace del dispositivo. Esto hace que aparezca una variación de la dirección de la capa de enlace dentro de la dirección IPv6.

Configuración de dirección IPv6

Al igual que IPv4, la dirección IPv6 se puede asignar de forma manual o automática. Sin embargo, IPv6 tiene dos tipos diferentes de configuración automatizada, DHCPv6 y *Stateless Address Autoconfiguration* (SLAAC)

SLAAC es el más fácil de los dos métodos automatizados y está integrado en el estándar IPv6. Los mensajes usan el nuevo *Neighbor Discovery Protocol* que permite que los dispositivos se encuentren y soliciten información sobre una red. Esta información es enviada por enrutadores y puede contener prefijos IPv6 que los dispositivos pueden usar combinándolos con un identificador de interfaz de su elección siempre que la dirección resultante aún no esté en uso. Los dispositivos no proporcionan comentarios al enrutador sobre las direcciones reales que han creado.

Por otro lado, DHCPv6 es el DHCP actualizado hecho para trabajar con los cambios de IPv6. Permite un control más preciso sobre la información entregada a los clientes, como permitir que la misma dirección se entregue al mismo cliente cada vez que se conecta y enviar más opciones al cliente que SLAAC. Con DHCPv6, los clientes necesitan obtener el consentimiento explícito de un servidor DHCP para usar una dirección.

El método para asignar manualmente una dirección IPv6 a una interfaz es el mismo que con IPv4:

```
$ sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev ens33
```

Para verificar que la asignación ha funcionado se usa el mismo comando `ip addr show`:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
    inet6 2001:db8:0:abcd::7334/64 scope global
       valid_lft forever preferred_lft forever
```

Aquí también vemos la dirección de enlace local `fe80::010c:29ff:fe33:3b25/64`.

Al igual que IPv4, el comando `ping` también se puede usar para confirmar la accesibilidad de los dispositivos a través de IPv6:

```
$ ping 2001:db8:0:abcd::1
PING 2001:db8:0:abcd::1(2001:db8:0:abcd::1) 56 data bytes
64 bytes from 2001:db8:0:abcd::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=3 ttl=64 time=0.072 ms

--- 2001:db8:0:abcd::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 43ms
rtt min/avg/max/mdev = 0.030/0.047/0.072/0.018 ms
```

NOTE En algunos sistemas Linux, `ping` no es compatible con IPv6. Estos sistemas proporcionan un comando dedicado `ping6` en su lugar.

Para nuevamente verificar la dirección del enlace local, use `ping` de nuevo. Pero como todas las interfaces usan el prefijo `fe80::/64`, se debe especificar la interfaz correcta junto con la dirección:


```
$ ping6 -c 1 fe80::010c:29ff:fe33:3b25%ens33
PING fe80::010c:29ff:fe33:3b25(fe80::010c:29ff:fe33:3b25) 56 data bytes
64 bytes from fe80::010c:29ff:fe33:3b25%ens33: icmp_seq=1 ttl=64 time=0.049 ms

--- fe80::010c:29ff:fe33:3b25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

DNS

Las direcciones IP son difíciles de recordar y no tienen exactamente un alto factor de frialdad si está tratando de comercializar un servicio o producto. Aquí es donde entra en juego el *Domain Name System*. En su forma más simple, DNS es una guía telefónica distribuida que asigna nombres de dominio fáciles de recordar como `example.com` a direcciones IP. Cuando, por ejemplo, un usuario navega a un sitio web, ingresa el nombre de host DNS como parte de la URL. El navegador web luego envía el nombre DNS a la resolver DNS que se haya configurado. Este resolver a su vez encontrará la dirección que se correlaciona con el dominio. Luego, el resolver responde con esa dirección y el navegador web intenta llegar al servidor web en esa dirección IP.

Los resolvers que Linux usa para buscar datos DNS se configuran en el archivo de configuración `/etc/resolv.conf`:

```
$ cat /etc/resolv.conf
search lpi
nameserver 192.168.0.1
```

Cuando el resolver realiza una búsqueda de nombre, primero verifica el archivo `/etc/hosts` para ver si contiene una dirección para el nombre solicitado. Si lo hace, devuelve esa dirección y no se pone en contacto con el DNS. Esto permite a los administradores de red proporcionar resolución de nombres sin tener que pasar por el esfuerzo de configurar un servidor DNS completo. Cada línea en ese archivo contiene una dirección IP seguida de uno o más nombres:

```
127.0.0.1          localhost.localdomain   localhost
::1                localhost.localdomain   localhost
192.168.0.10       server
2001:db8:0:abcd::f server
```

Para realizar una búsqueda en el DNS use el comando `host`:

```
$ host learning.lpi.org
learning.lpi.org has address 208.94.166.198
```

Se puede recuperar información más detallada utilizando el comando `dig`:

```
$ dig learning.lpi.org
; <<>> DiG 9.14.3 <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2ac55879b1adef30a93013705d3306d2128571347df8eadf (bad)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   550 IN  A   208.94.166.198

;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sat Jul 20 14:20:21 EST 2019
;; MSG SIZE  rcvd: 89
```

Aquí también vemos el nombre de los tipos de registro DNS, en este caso `A` para IPv4.

Sockets

Un *socket* es un punto final de comunicación para dos programas que se comunican entre sí. Si el socket está conectado a una red, los programas pueden ejecutarse en diferentes dispositivos, como un navegador web que se ejecuta en la computadora portátil de un usuario y un servidor web que se ejecuta en el centro de datos de una empresa.

Hay tres tipos principales de sockets:

Unix Sockets
: Son procesos de conexión que se ejecutan en el mismo dispositivo.

UDP (User Datagram Protocol) Sockets

Se conectan aplicaciones usando un protocolo que es rápido pero no resistente.

TCP (Transmission Control Protocol) Sockets

Son más confiables que los sockets UDP y confirman la recepción de datos.

Los sockets Unix solo pueden conectar aplicaciones que se ejecutan en el mismo dispositivo. Sin embargo, los sockets TCP y UDP pueden conectarse a través de una red. TCP permite una secuencia de datos que siempre llega en el orden exacto en que se envió. UDP es más fugaz; el paquete se envía pero no se garantiza su entrega en el otro extremo. Sin embargo, UDP carece de la sobrecarga de TCP, por lo que es perfecto para aplicaciones de baja latencia, como los videojuegos en línea.

TCP y UDP usan puertos para abordar múltiples sockets en la misma dirección IP. Si bien el puerto de origen para una conexión suele ser aleatorio, los puertos de destino están estandarizados para un servicio específico. Por ejemplo HTTP generalmente está alojado en el puerto 80, HTTPS se ejecuta en el puerto 443. SSH un protocolo para iniciar sesión de forma segura en un sistema Linux remoto, escucha en el puerto 22.

El comando `ss` le permite a un administrador investigar todos los sockets en una computadora Linux. Muestra todo, desde la dirección de origen, la dirección de destino y el tipo. Una de sus mejores características es el uso de filtros para que un usuario pueda monitorear los sockets en cualquier estado de conexión que desee. El comando `ss` se puede ejecutarse con un conjunto de opciones, así como una expresión de filtro para limitar la información que se muestra.

Cuando se ejecuta sin ninguna opción, el comando muestra una lista de todos los sockets establecidos. El uso de la opción `-p` incluye información sobre el proceso con cada socket. La opción `-s` muestra un resumen de los sockets. Hay muchas más opciones disponibles para esta herramienta, pero el último conjunto de las principales son `-4` y `-6` para reducir el protocolo IP a IPv4 o IPv6 respectivamente, `-t` y `-u` permiten al administrador seleccionar sockets TCP o UDP y `-l` para mostrar solo los sockets que escuchan nuevas conexiones.

Por ejemplo, el siguiente comando enumera todos los sockets TCP actualmente en uso:

```
$ ss -t
State       Recv-Q  Send-Q    Local Address:Port      Peer Address:Port
ESTAB       0       0           192.168.0.5:49412      192.168.0.1:https
ESTAB       0       0           192.168.0.5:37616      192.168.0.1:https
ESTAB       0       0           192.168.0.5:40114      192.168.0.1:https
ESTAB       0       0           192.168.0.5:54948      192.168.0.1:imap
...
```

Ejercicios guiados

1. Se le solicita a un ingeniero de red que asigne dos direcciones IP a la interfaz `ens33` de un host, una dirección IPv4 (`192.168.10.10/24`) y una dirección IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). ¿Qué comandos debe ingresar para lograr esto?

2. ¿Qué direcciones de la lista son privadas?



`192.168.10.1`	
`120.56.78.35`	
`172.16.57.47`	
`10.100.49.162`	
`200.120.42.6`	

3. ¿Qué entrada agregaría al archivo hosts para asignar `192.168.0.15` a `example.com`?

4. ¿Qué efecto tendría el siguiente comando?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```


Ejercicios exploratorios

1. Nombre el tipo de registro DNS utilizado para atender las siguientes solicitudes:
   - Dato textual
   - Búsqueda inversa de dirección IP
   - Un dominio no tiene dirección propia y depende de otro dominio para esta información
   - Servidor de correo
2. Linux tiene una característica llamada puente (bridging). ¿Qué hace y cómo es útil?
3. ¿Qué opción se debe proporcionar al comando `ss` para ver todos los sockets UDP establecidos?
4. ¿Qué comando muestra un resumen de todos los sockets que se ejecutan en un dispositivo Linux?
5. El siguiente resultado es generado por el comando del ejercicio anterior. ¿Cuántos sockets TCP y UDP están activos?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*     0         -         -
RAW   1         0         1
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

Resumen

Este tema cubrió la conexión en red de su computadora Linux. Primero aprendimos sobre los diversos niveles de redes:

- La capa de enlace conecta los dispositivos directamente.
- La capa de red que proporciona enrutamiento entre redes y un espacio de direcciones global.
- La capa de aplicación donde las aplicaciones se conectan entre sí.

Hemos visto cómo se usan IPv4 e IPv6 para abordar computadoras individuales y cómo TCP y UDP enumeran los sockets utilizados por las aplicaciones para conectar entre sí. También aprendimos cómo se usa DNS para resolver nombres en direcciones IP.

Comandos utilizados en los ejercicios:

`dig`
: Consulta información de DNS y proporcionar información detallada sobre las consultas y respuestas del mismo.

`host`
: Consulta información DNS y proporcionando salida.

`ip`
: Configura las redes en Linux, incluidas las interfaces de red, las direcciones y el enrutamiento.

`ping`
: Prueba la conectividad con un dispositivo remoto.

`ss`
: Muestra información sobre sockets.

Respuestas a los ejercicios guiados

1. Se le solicita a un ingeniero de red que asigne dos direcciones IP a la interfaz `ens33` de un host, una dirección IPv4 (`192.168.10.10/24`) y una dirección IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). ¿Qué comandos debe ingresar para lograr esto?

```
sudo ip addr add 192.168.10.10/24 dev ens33
sudo ip addr add 2001:0:0:abcd:0:8a2e:0370:7334/64 dev ens33
```

2. ¿Qué direcciones de la lista son privadas?

`192.168.10.1`	X
`120.56.78.35`	
`172.16.57.47`	X
`10.100.49.162`	X
`200.120.42.6`	

3. ¿Qué entrada agregaría al archivo hosts para asignar `192.168.0.15` a `example.com`?

```
192.168.0.15  example.com
```

4. ¿Qué efecto tendría el siguiente comando?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Agregaría una ruta predeterminada a la tabla que envía todo el tráfico IPv6 al enrutador con la dirección interna `2001:db8:0:abcd::1`.

Respuestas a los ejercicios exploratorios

1. Nombre el tipo de registro DNS utilizado para atender las siguientes solicitudes:
   - Dato textual

     TXT
   - Búsqueda inversa de dirección IP

     PTR
   - Un dominio no tiene dirección propia y depende de otro dominio para esta información

     CNAME
   - Servidor de correo

     MX
2. Linux tiene una característica llamada puente (bridging). ¿Qué hace y cómo es útil?

   Un puente conecta múltiples interfaces de red. Todas las interfaces conectadas a un puente pueden comunicarse como si estuvieran conectadas a la misma red de capa de enlace: todos los dispositivos usan direcciones IP de la misma subred y no requieren un enrutador para conectarse entre sí.
3. ¿Qué opción se debe proporcionar al comando `ss` para ver todos los sockets UDP establecidos?

   La opción `-u` muestra todos los sockets UDP establecidos.

   ```
   ¿Qué comando muestra un resumen de todos los sockets que se ejecutan en un
   dispositivo Linux?
   ```

   El comando `ss -s` muestra un resumen de todos los sockets.
4. El siguiente resultado es generado por el comando del ejercicio anterior. ¿Cuántos sockets TCP y UDP están activos?

```
Total: 978 (kernel 0)
TCP:   4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*     0         -         -
RAW   1         0         1
UDP   7         5         2
TCP   4         3         1
INET      12        8         4
FRAG      0         0         0
```

11 sockets TCP y UDP están activos.



Tema 5: Seguridad y sistema de permisos de archivos



5.1 Seguridad básica e identificación de tipos de usuarios

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 5.1

Importancia

2

Áreas de conocimiento clave

- Usuario root y usuarios estándar
- Usuarios de sistema

Lista parcial de archivos, términos y utilidades

- `/etc/passwd`, `/etc/shadow`, `/etc/group`
- `id`, `last`, `who`, `w`
- `sudo`, `su`



5.1 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	5 Seguridad y permisos de archivos
Objetivo:	5.1 Seguridad básica e identificación de tipos de usuario
Lección:	1 de 1

Introducción

Esta lección se centrará en la terminología básica de las cuentas, los controles de acceso y su seguridad local de los sistemas Linux; además de las herramientas que provee la interfaz de línea de comandos (CLI), en un sistema Linux para controles básicos de acceso de seguridad y los archivos básicos para admitir cuentas de usuarios y grupos, incluyendo los utilizados para escalar privilegios.

La seguridad básica en los sistemas Linux se basa en los controles de acceso de Unix, que a pesar de tener casi cincuenta años, son bastante efectivos en comparación con algunos otros sistemas operativos de un linaje mucho más nuevo. Incluso algunos sistemas operativos populares basados en Unix tienden a "tomar libertades" que se centran en la "facilidad de acceso", mientras que Linux no lo hace.

Los modernos entornos e interfaces de escritorio en Linux simplifican la creación y administración de los usuarios y a menudo, automatizan la asignación de controles de acceso cuando un usuario inicia sesión, por ejemplo, en la pantalla, la configuración audio y otros servicios, ya que prácticamente no requiere intervención manual del administrador del sistema. Sin embargo, es importante comprender los conceptos básicos de un subyacente sistema operativo Linux.

Cuentas

La seguridad implica muchos conceptos, uno de los más comunes es el concepto general de controles de acceso. Antes de poder abordar los controles de acceso, en los archivos (como la propiedad y los permisos) se debe comprender los conceptos básicos de las cuentas de usuario en Linux, que se dividen en varios tipos.

Cada usuario en un sistema Linux tiene una cuenta asociada que además de la información de inicio de sesión (como nombre de usuario y contraseña) también define cómo y dónde puede interactuar el usuario con el sistema. Los privilegios y los controles de acceso definen los "límites" dentro de los cuales puede operar cada usuario.

Identificadores (UIDs/GIDs)

Los *User* y *Group Identifiers* (UIDs/GIDs) son las referencias básicas y enumeradas a las cuentas. Las primeras implementaciones eran enteros limitados a 16 bits (valores de 0 a 65535), pero los sistemas del siglo XXI admiten UIDs y GIDs de 64 bits. Los usuarios y grupos se enumeran de forma independiente, por lo que el mismo ID puede representar tanto a un usuario como a un grupo.

Cada usuario no tiene solo un UID, sino también un *GID primario*. El GID primario para un usuario puede ser exclusivo de solo ese usuario y puede terminar sin ser utilizado por ningún otro. Sin embargo, este grupo también podría ser un grupo compartido por numerosos usuarios. Además de estos grupos principales, cada usuario también puede ser miembro de otros grupos.

Por defecto en los sistemas Linux, cada usuario está asignado a un grupo con el mismo nombre de usuario y el mismo GID que su UID. Por ejemplo, si crea un nuevo usuario llamado `newuser`, por defecto, su grupo predeterminado también será `newuser`.

La cuenta de superusuario

En Linux, la cuenta de superusuario es `root`, que siempre tiene el UID 0. El superusuario a veces se denomina administrador del sistema y tiene acceso y control ilimitados sobre el sistema, incluidos otros usuarios.

El grupo predeterminado para el superusuario tiene el GID `0` y también se denomina `root`. El directorio de inicio para el superusuario es un directorio dedicado de nivel superior, `/root`, al que solo puede acceder el usuario `root`.

Cuentas de usuario estándar (User Accounts)

Todas las cuentas que no sean `root` son cuentas de usuario técnicamente regulares, pero en un sistema Linux el término coloquial *user account* a menudo significa una cuenta de usuario "regular"

(sin privilegios). Por lo general, tienen las siguientes propiedades, con algunas excepciones:

- El UID comienzan en 1000 (4 dígitos), aunque algunos sistemas heredados pueden comenzar en 500.
- Posee un definido directorio de inicio, generalmente es un subdirectorio de `/home`, dependiendo de la configuración local del sitio.
- Posee un Shell definido para el inicio de sesión. En Linux, el shell predeterminado suele ser *Bourne Again Shell* (`/bin/bash`), aunque puede haber otros disponibles.

Si una cuenta de usuario no tiene un shell válido en sus atributos, el usuario no podrá abrir un shell interactivo. Usualmente `/sbin/nologin` se usa como un shell inválido. Esto puede tener un propósito, solo si el usuario autenticará para otros servicios que no sean la consola o el acceso SSH, por ejemplo, solo para el acceso Secure FTP (`sftp`).

> **NOTE** Para evitar confusiones, el término *user account* solo se aplicará a las cuentas de usuario estándar o regulares en adelante. Por ejemplo, *system account* se usará para explicar una cuenta de Linux que sea de tipo usuario del sistema.

Cuentas del sistema (System Accounts)

Las *cuentas del sistema (System accounts)* normalmente se crean en el momento de la instalación del sistema. Estos son para instalaciones, programas y servicios que no se ejecutarán como superusuario. En un mundo ideal, todas estas serían instalaciones del sistema operativo.

Las cuentas del sistema varían, pero sus atributos incluyen:

- Los UID suelen ser inferiores a 100 (2 dígitos) o 500-1000 (3 dígitos).
- Ya sea o *no* que posean un directorio de inicio dedicado o un directorio que generalmente no es subdirectorio de `/home`.
- Sin shell de inicio de sesión válido (normalmente `/sbin/nologin`), con raras excepciones.

En Linux la mayoría de las cuentas del sistema nunca iniciarán sesión y tampoco necesitan un shell en sus atributos. Muchos procesos de propiedad y ejecutados por las cuentas del sistema se bifurcan en su propio entorno por la administración del sistema, ejecutándose con la cuenta del sistema especificada. Estas cuentas generalmente tienen privilegios limitados o no tienen privilegios (la mayoría de las veces).

> **NOTE** Desde el punto de vista de LPI Linux Essentials, las cuentas del sistema son UID < 1000, con UID de 2 o 3 dígitos y un GID.

En general, las cuentas del sistema no deben tener un válido shell de inicio de sesión. Lo contrario sería un problema de seguridad en la mayoría de los casos.

Cuentas de servicio (Service Accounts)

Las *cuentas de servicio* generalmente se crean cuando los servicios se instalan y configuran. Al igual que las cuentas del sistema, son para instalaciones, programas y servicios que no se ejecutarán como superusuario.

Las cuentas de sistema y servicio son similares y se intercambian a menudo. Esto incluye la ubicación de los directorios de inicio que normalmente están fuera de `/home`, si se define en todas (las cuentas de servicio a menudo tienen más probabilidades de tener una ubicación, en comparación a las cuentas del sistema), además no hay un shell válido de inicio de sesión. Aunque no existe una definición estricta, la diferencia principal entre las cuentas de sistema y servicio se desglosa en UID/GID.

Cuenta del sistema (System Account)

UID/GID <100 (2- dígitos) o <500-1000 (3- dígitos)

Cuenta de servicio (Service account)

UID/GID >1000 (4+ dígitos), pero no una cuenta de usuario "estándar" o "regular", una cuenta para servicios con un UID/GID >1000 (4+ dígitos)

Algunas distribuciones de Linux todavía tienen cuentas de servicio previamente reservadas con un UID <100, y también podrían considerarse una cuenta del sistema, aunque no se creen en la instalación del sistema operativo. Por ejemplo, en las distribuciones Linux basadas en Fedora (incluido Red Hat), el usuario Apache del servidor web tiene un UID (y GID) 48. Claramente es una cuenta del sistema, a pesar de tener un directorio de inicio (generalmente en `/usr/share/httpd` o `/var/www/html/`).

NOTE Desde el punto de vista de LPI Linux Essentials, las cuentas del sistema son UID <1000, y las cuentas de usuario normales son UID >1000. Como las cuentas de usuario normales son >1000, estos UID también pueden incluir cuentas de servicio.

Shells de inicio de sesión y directorios de inicio

Algunas cuentas tienen un shell de inicio de sesión, mientras que otras no, ya que no requieren acceso interactivo y por temas de seguridad. En la mayoría de las distribuciones de Linux, el shell predeterminado de inicio de sesión es *Bourne Again Shell* o `bash`, pero puede haber otros shells disponibles, como C Shell (`csh`), Korn shell (`ksh`) o Z shell (`zsh`), entre otros.

Un usuario puede cambiar su shell de inicio de sesión utilizando el comando `chsh`. Por defecto, el comando se ejecuta en modo interactivo y muestra un mensaje preguntando qué shell debe usar. La respuesta debería ser la ruta completa del binario de shell, como se muestra a continuación:

```
$ chsh

Changing the login shell for emma
Enter the new value, or press ENTER for the default
    Login Shell [/bin/bash]: /usr/bin/zsh
```

También puede ejecutar el comando en modo no interactivo, con el parámetro `-s` seguido de la ruta del binario, algo así:

```
$ chsh -s /usr/bin/zsh
```

La mayoría de las cuentas tienen definido un directorio de inicio. En Linux, este suele ser la única ubicación donde esa cuenta de usuario tiene acceso garantizado de escritura, con algunas excepciones (por ejemplo, áreas del sistema de archivos temporales). Sin embargo, por razones de seguridad, algunas cuentas se configuran a propósito para que no tengan acceso de escritura ni siquiera a su propio directorio de inicio.

Obtenga información sobre sus usuarios

Enumerar la información básica del usuario es una práctica común y cotidiana en un sistema Linux. En algunos casos, se deberá cambiar de usuario y aumentar los privilegios para completar algunas tareas.

Incluso los usuarios tienen la capacidad de enumerar atributos y acceder desde la línea de comandos utilizando los comandos a continuación. La información básica en un contexto limitado no es una operación privilegiada.

Listar la información actual de un usuario en la línea de comando es tan simple como un comando de dos letras, `id`. La salida variará según su ID de inicio de sesión:

```
$ id
uid=1024(emma) gid=1024(emma)
1024(emma),20(games),groups=10240(netusers),20480(netadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

En la lista anterior, el usuario (`emma`) tiene identificadores que se desglosan de la siguiente manera:

- `1024` = ID de usuario (UID), seguido del nombre de usuario (nombre común también conocido como nombre de inicio de sesión) entre paréntesis.
- `1024` = ID del grupo *primario* (GID), seguido del nombre de grupo (nombre común) entre paréntesis.
- Una lista de GID adicionales (nombres de grupo) a los que también pertenece el usuario.

Para enumerar la última vez que los usuarios han iniciado sesión en el sistema se utiliza el comando `last`:

```
$ last
emma     pts/3        ::1              Fri Jun 14 04:28   still logged in
reboot   system boot  5.0.17-300.fc30. Fri Jun 14 04:03   still running
reboot   system boot  5.0.17-300.fc30. Wed Jun  5 14:32 - 15:19  (00:46)
reboot   system boot  5.0.17-300.fc30. Sat May 25 18:27 - 19:11  (00:43)
reboot   system boot  5.0.16-100.fc28. Sat May 25 16:44 - 17:06  (00:21)
reboot   system boot  5.0.9-100.fc28.x Sun May 12 14:32 - 14:46  (00:14)
root     tty2                          Fri May 10 21:55 - 21:55  (00:00)
    ...
```

La información que figura en las columnas puede variar, pero algunas entradas notables en el listado anterior son:

- Un usuario (`emma`) inició sesión a través de la red (pseudo TTY `pts/3`) y todavía está conectado.
- Se enumera la hora del arranque actual, junto con el kernel. En el ejemplo anterior, unos 25 minutos antes de que el usuario inicie sesión.
- El superusuario (`root`) inició sesión a través de una consola virtual (TTY `tty2`), a mediados de mayo.

Una variante de `last` es el comando `lastb`, que enumera todos los últimos intentos de inicio de sesión incorrectos.

Los comandos `who` y `w` enumeran solo los inicios de sesión activos en el sistema:

```
$ who
emma  pts/3        2019-06-14 04:28 (::1)

$ w
 05:43:41 up  1:40,  1 user,  load average: 0.25, 0.53, 0.51
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
emma  pts/3     04:28    1:14m  0.04s  0.04s -bash
```

Ambos comandos enumeran parte de la misma información. Por ejemplo, el usuario (`emma`) ha iniciado sesión con un dispositivo pseudo TTY (`pts/3`) y la hora de inicio de sesión fue a las 04:28.

El comando `w` muestra más información, incluida la siguiente:

- La hora actual y cuánto tiempo ha estado funcionando el sistema
- ¿Cuántos usuarios están conectados?
- Los *promedios de carga (load averages)* de los últimos 1, 5 y 15 minutos

Y la información adicional para cada sesión de un usuario activo.

- Seleccionar, tiempos totales de utilización de la CPU (`IDLE`,`JCPU` y `PCPU`)
- El proceso actual (`-bash`). El tiempo total de utilización de la CPU de ese proceso es el último elemento (`PCPU`).

Ambos comandos tienen más opciones para enumerar información adicional.

Cambio de usuarios y aumento de privilegios

En un mundo ideal, los usuarios nunca necesitarían escalar privilegios para completar sus tareas. El sistema "simplemente funcionaría" y todo estaría configurado para varios accesos.

Afortunadamente para nosotros, Linux está listo para usarse, funciona así para la mayoría de los usuarios que no son administradores del sistema, a pesar de seguir siempre un modelo de seguridad de privilegios mínimos.

Sin embargo, hay comandos que permiten escalar de privilegios cuando es necesario. Dos de los más importantes son `su` y `sudo`.

En la mayoría de los sistemas Linux actuales, el comando `su` solo se usa para escalar privilegios a root, que es el usuario predeterminado si no se especifica un nombre de usuario después del nombre del comando. Si bien se puede usar para cambiar a otro usuario, no es una buena práctica: los

usuarios deben iniciar sesión desde otro sistema, a través de la red, consola física o terminal en el sistema.

```
emma ~$ su -
Password:
root ~#
```

Después de ingresar la contraseña de superusuario (`root`), el usuario tiene un shell de superusuario (observe el `#` al final del símbolo del sistema) y es para todos los efectos, el superusuario (`root`).

Compartir contraseñas es una práctica de seguridad muy mala, por lo que debe haber muy pocas, si es que hay alguna necesidad de usar el comando `su` en un sistema Linux moderno.

El símbolo de dólar ($) indica en la línea de comandos un shell de usuario no privilegiado, mientras que el símbolo de libra (#) indica en la línea de comandos un shell de superusuario (`root`). Se recomienda encarecidamente que el carácter final de cualquier aviso *nunca* se cambie de este estándar de "comprensión universal", ya que esta nomenclatura se utiliza en materiales de aprendizaje, incluidos estos.

WARNING Nunca cambie al superusuario (`root`) sin pasar el parámetro de inicio de sesión (`-`). A menos que el sistema operativo o el proveedor de software indiquen lo contrario cuando se requiere `su`, ejecute siempre `su -` con excepciones extremadamente limitadas. Los entornos de usuario pueden causar cambios y problemas de configuración no deseados cuando se usan en modo completo de privilegio como superusuario.

¿Cuál es el mayor problema con el uso de `su` para cambiar al superusuario (`root`)? Si la sesión de un usuario normal se ha visto comprometida, se podría capturar la contraseña de superusuario (`root`). Ahí es donde entra en juego el “Switch User Do'' (o "`Superuser Do”):

```
$ cat /sys/devices/virtual/dmi/id/board_serial
cat: /sys/devices/virtual/dmi/id/board_serial: Permission denied

$ sudo cat /sys/devices/virtual/dmi/id/board_serial
[sudo] password for emma:
/6789ABC/
```

En la lista anterior, el usuario está intentando buscar el número de serie de su placa del sistema. Sin embargo, el permiso es denegado, ya que esa información está marcada como privilegiada.

Sin embargo, al usar `sudo`, el usuario ingresa su propia contraseña para autenticar quién es. Si ha sido autorizado en la configuración de `sudoers` para ejecutar ese comando con privilegio, con las opciones permitidas, funcionará.

TIP Por defecto, el primer comando `sudo` autorizado autenticará los siguientes comandos `sudo` por un período de tiempo (muy corto). Esto es configurable por el administrador del sistema.

Archivos de control de acceso

Casi todos los sistemas operativos tienen un conjunto de ubicaciones utilizados para almacenar controles de acceso. En Linux, estos son típicamente archivos de texto ubicados en el directorio `/etc`, que es donde deben almacenarse los archivos de configuración del sistema. Por defecto, todos los usuarios del sistema pueden leer este directorio, pero solo `root` puede escribirlo.

Los archivos principales relacionados con cuentas de usuario, atributos y control de acceso son:

`/etc/passwd`
: Este archivo almacena información básica sobre los usuarios en el sistema, incluyendo UID y GID, directorio de inicio, tipo de shell, etc. A pesar del nombre, aquí no se almacenan contraseñas.

`/etc/group`
: Este archivo almacena información básica sobre todos los grupos de usuarios en el sistema, como el nombre del grupo, GID y sus miembros.

`/etc/shadow`
: Aquí es donde se almacenan las contraseñas de los usuarios. Por seguridad son hash.

`/etc/gshadow`
: Este archivo almacena información más detallada sobre los grupos, incluida una contraseña cifrada que permite a los usuarios convertirse temporalmente en un miembro del grupo. En una lista de usuarios puede convertirse en un miembro de inclusive una lista de administradores.

WARNING Estos archivos no están diseñados y nunca deben editarse directamente. Esta lección solo cubre la información almacenada en estos archivos y no edita estos archivos.

Por defecto, cada usuario puede ingresar a `/etc` y leer los archivos `/etc/passwd` y `/etc/group`. También de manera predeterminada, ningún usuario, excepto `root`, puede leer los archivos

`/etc/shadow` o `/etc/gshadow`.

También hay archivos relacionados con la escalada de privilegios básicos en sistemas Linux, como en los comandos `su` y `sudo`. Por defecto, estos solo son accesibles por el usuario `root`.

`/etc/sudoers`
: Este archivo controla quién puede y cómo usar el comando `sudo`.

`/etc/sudoers.d`
: Este directorio puede contener archivos que complementan la configuración del archivo `sudoers`.

Desde el punto de vista para el examen de LPI Linux Essentials, solo debe conocer la ruta y el nombre del archivo de configuración de sudo, `/etc/sudoers`. Su configuración está más allá del alcance de estos materiales.

WARNING	Aunque `/etc/sudoers` es un archivo de texto, nunca debe editarse directamente. Si necesita cambiar su contenido, debe realizar utilizar la utilidad `visudo`.

/etc/passwd

El archivo `/etc/passwd` se conoce comúnmente como el "archivo de contraseña". Cada línea contiene múltiples campos delimitados por dos puntos (`:`). A pesar del nombre, actualmente el hash real de contraseña no se almacena en este archivo.

La sintaxis típica de una línea en este archivo es la siguiente:

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
```

Donde:

`USERNAME`
: El nombre de usuario conocido como login (nombre), como `root`, `nobody`, `emma`.

`PASSWORD`
: Ubicación heredada del hash de contraseña. Casi siempre `x`, lo que indica que la contraseña está almacenada en el archivo `/etc/shadow`.

`UID`
: El ID del usuario (UID), como `0`, `99`, `1024`.

`GID`
: El ID del grupo (GID), como `0`, `99`, `1024`.

`GECOS`
: Una lista CSV de información del usuario que incluye nombre, ubicación, número de teléfono. Por ejemplo: `Emma Smith, 42 Douglas St, 555.555.5555`.

`HOMEDIR`
: Ruta del directorio de inicio del usuario, como `/root`, `/home/emma`, etc.

`SHELL`
: El shell predeterminado para esta usuario, como `/bin/bash`, `/sbin/nologin`, `/bin/ksh`, etc.

Por ejemplo, la siguiente línea describe al usuario `emma`:

```
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

Comprendiendo el campo GECOS

El campo GECOS contiene tres (3) o más campos delimitados por una coma (`,`), también conocida como una lista de *valores separados por comas* (CSV). Aunque no existe un estándar obligatorio, los campos suelen estar en el siguiente orden:

```
NAME,LOCATION,CONTACT
```

Donde:

`NAME`
: Es el “Nombre completo” (“Full Name”) del usuario o el “Nombre del software” (“Software Name”) en el caso de una cuenta de servicio.

`LOCATION`
: Suele ser la ubicación física del usuario dentro de un edificio, número de habitación, el departamento de contacto o persona en el caso de una cuenta de servicio.

`CONTACT`

Enumera información de contacto, como el número de teléfono del hogar o del trabajo.

Los campos adicionales pueden incluir información de contacto adicional, como un número de casa o una dirección de correo electrónico. Para cambiar la información en el campo GECOS, use el comando `chfn` y responda las preguntas, como se verá en el siguiente ejemplo. Si no se proporciona un nombre de usuario después del nombre del comando, cambiará la información para el usuario actual:

```
$ chfn

Changing the user information for emma
Enter the new value, or press ENTER for the default
    Full Name: Emma Smith
    Room Number []: 42
    Work Phone []: 555.555.5555
    Home Phone []: 555.555.6666
```

/etc/group

El archivo `/etc/group` contiene campos siempre delimitados por dos puntos (`:`) que almacenan información básica sobre los grupos en el sistema. A veces se le llama "archivo de grupo". La sintaxis para cada línea es:

```
NAME:PASSWORD:GID:MEMBERS
```

Donde:

`NAME`

Es el nombre del grupo, como `root`, `users`, `emma`, etc.

`PASSWORD`

Ubicación heredada de un hash de contraseña de un grupo opcional. Casi siempre `x`, lo que indica que la contraseña (si está definida) y se almacena en el archivo `/etc/gshadow`.

`GID`

El ID del grupo (GID), como `0`, `99`, `1024`.

`MEMBERS`
: Una lista de nombres de usuario separados por comas que son miembros del grupo, como `jsmith,emma`.

El siguiente ejemplo muestra una línea que contiene información sobre el grupo `students`:

```
students:x:1023:jsmith,emma
```

No es necesario que el usuario aparezca en el campo de miembros cuando el grupo es el primario para un usuario. Si un usuario está en la lista, entonces es redundante, es decir, no hay cambio en la funcionalidad, listada o no.

> **NOTE** El uso de contraseñas para grupos está más allá del alcance de esta sección, sin embargo, si se define, el hash de la contraseña se almacena en el archivo `/etc/gshadow`. Esto también está más allá del alcance de esta sección.

/etc/shadow

La siguiente tabla enumera los atributos almacenados en el archivo `/etc/shadow`, comúnmente conocido como el "shadow file". El archivo contiene campos siempre delimitados por dos puntos (`:`). Aunque el archivo tiene muchos campos, la mayoría están más allá del alcance de esta lección, aparte de los dos primeros.

La sintaxis básica para una línea en este archivo es:

```
USERNAME:PASSWORD:LASTCHANGE:MINAGE:MAXAGE:WARN:INACTIVE:EXPDATE
```

Donde:

`USERNAME`
: El nombre de usuario (igual que `/etc/passwd`), como `root`, `nobody`, `emma`.

`PASSWORD`
: Un hash unidireccional de la contraseña, precedido por un "salt". Por ejemplo: `!!`, `!$1$01234567$ABC…`, `$6$012345789ABCDEF$012…`.

`LASTCHANGE`
: Fecha del último cambio de contraseña en días desde la "época", como `17909`.

`MINAGE`

Minimo de días antes que al usuario se le permite cambiar la contraseña..

`MAXAGE`

Máximo de días antes que al usuario se le permite cambiar la contraseña

`WARN`

Período de advertencia (en días) antes de que caduque la contraseña.

`INACTIVE`

Antigüedad máxima (en días) de la contraseña después del vencimiento.

`EXPDATE`

Fecha de caducidad (en días) de la contraseña desde la “época”.

En el siguiente ejemplo, puede ver una entrada de muestra del archivo `/etc/shadow`. Tenga en cuenta que algunos valores, como `INACTIVE` y `EXPDATE` no están definidos.

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmjlIwgu7hGjH/eytSdttbmVv0MlyTMFgBIXESFNUm
To9EGxxH1OT1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

La “época” de un sistema POSIX es la medianoche (0000), hora universal coordinada (UTC), el jueves 1 de enero de 1970. La mayoría de las fechas y horas POSIX están en segundos desde “época” o en el caso del archivo `/etc/shadow`, en días desde la “época”.

> **NOTE** El archivo shadow está diseñado para que solo el superusuario pueda leerlo, y seleccione los servicios de autenticación del sistema central que verifican el hash de contraseña unidireccional al iniciar sesión u otro tiempo de autenticación.

Aunque existen diferentes soluciones de autenticación, el método elemental de almacenamiento de contraseñas es la *función hash* unidireccional. Esto se hace para que la contraseña nunca se almacene en texto sin cifrar en un sistema, ya que la función de hash no es reversible. Puede convertir una contraseña en un hash, pero (idealmente) no es posible volver a convertir un hash en una contraseña.

Como máximo, se requiere un método de fuerza bruta para trocear todas las combinaciones de una contraseña, hasta que coincida. Para mitigar el problema de que un hash de contraseña sea descifrado en un sistema, los sistemas Linux usan un “salt” aleatoria en cada hash de contraseña para un usuario. Por lo tanto, el hash para una contraseña de usuario en un sistema Linux generalmente

no será el mismo que en otro sistema Linux, incluso si la contraseña es la misma.

En el archivo `/etc/shadow`, la contraseña puede tomar varias formas. Estos formularios generalmente incluyen lo siguiente:

`!!`

Esto significa una cuenta “deshabilitada” (sin posible autenticación) y sin una contraseña hash almacenada.

`!$1$01234567$ABC…`

Una cuenta “deshabilitada” (debido al signo de exclamación inicial), con una función hash anterior, hash salt y cadena hash almacenada.

`$1$0123456789ABC$012…`

Una cuenta “habilitada”, con una función hash, hash salt y cadena de hash almacenados.

La función hash, hash salt y la cadena hash están precedidas y delimitadas por un símbolo de dólar (`$`). La longitud del hash salt debe ser de entre ocho y dieciséis (8-16) caracteres. Ejemplos de los tres más comunes son los siguientes:

`$1$01234567$ABC…`

Una función hash de MD5 (`1`), con un ejemplo de longitud hash de ocho.

`$5$01234567ABCD$012…`

Una función hash de SHA256 (`5`), con un ejemplo de longitud hash de doce.

`$6$01234567ABCD$012…`

Una función hash de SHA512 (`6`), con un ejemplo de longitud hash de doce.

NOTE La función hash MD5 se considera criptográficamente insegura con el nivel actual de ASIC (2010s y posteriores) e incluso el rendimiento de SIMD de computación general. Por ejemplo, los Estándares Federales de Procesamiento de Información (FIPS) de USA. No permiten el uso de MD5 para ninguna función criptográfica, solo aspectos muy limitados de la validación, pero no la integridad de las firmas digitales o propósitos similares.

Desde el punto de vista de los objetivos y el examen de LPI Linux Essentials, solo comprenda que el hash de contraseña para un usuario local solo se almacena en el archivo `/etc/shadow` que solo selecciona, los servicios de autenticación pueden leer o el superusuario puede modificar a través de otros comandos .

Ejercicios guiados

1. Considere la siguiente salida del comando `id`:

   ```
   $ id emma
   uid=1000(emma) gid=1000(emma)
   groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
   ```

   ¿En qué archivos se almacenan los siguientes atributos?

UID y GID	
Grupos	

   - Además, ¿en qué archivo se almacena la contraseña del usuario?

2. ¿Cuál de los siguientes tipos de criptografía se usa de manera predeterminada para almacenar contraseñas localmente en un sistema Linux?
   - Asymmetric
   - One-way Hash
   - Symmetric
   - ROT13

3. Si una cuenta tiene un ID de usuario (UID) enumerada bajo 1000, ¿Qué tipo de cuenta es esta?

4. ¿Cómo puede obtener una lista de los inicios de sesión activos en su sistema y también un recuento de ellos?

5. Usando el comando `grep`, obtuvimos el resultado siguiente con la información sobre el usuario `emma`.

   ```
   $ grep emma /etc/passwd
   emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
   ```


Complete los espacios en blanco del gráfico con la información apropiada usando la salida del comando anterior.

Username	
Password	
UID	
Primary GID	
GECOS	
Home Directory	
Shell	

Ejercicios exploratorios

1. Compare los resultados de `last` con `w` y `who`. En comparación, ¿Qué detalles faltan a cada uno de los comandos ?

2. Intente emitir los siguientes comandos `who` y `w -his`.

   - ¿Qué información se ha eliminado de la salida del comando `w` con las opciones “no header” (`-h`) y “short” (`-s`)?

   - ¿Qué información se ha agregado en la salida del comando `w` con la opción “ip address” (`-i`)?

3. ¿Cúal archivo es el que almacena el hash de contraseña unidireccional de una cuenta de usuario?

4. ¿Qué archivo contiene la lista de grupos de los que es miembro una cuenta de usuario? ¿Qué lógica podría usarse para compilar una lista de grupos de los que es miembro una cuenta de usuario?

5. Por defecto uno o más (1+) de los siguientes archivos no son legibles por usuarios normales y sin privilegios. ¿Cuáles son?

   - `/etc/group`
   - `/etc/passwd`
   - `/etc/shadow`
   - `/etc/sudoers`

6. ¿Cómo puede cambiar el shell de inicio de sesión del usuario actual al Korn Shell (`/usr/bin/ksh`) en modo no interactivo?

7. ¿Por qué el directorio de inicio del usuario `root` no está ubicado dentro del directorio `/home`?

Resumen

En esta lección hemos descubierto las bases de datos de usuarios y grupos de Linux. Hemos aprendido las propiedades más importantes de los usuarios y grupos, incluidos sus nombres y sus ID numéricos. También hemos investigado cómo funciona el hashing de contraseñas en Linux y cómo se asignan los usuarios a los grupos.

Toda esta información se almacena en los siguientes cuatro archivos, que proporcionan los controles de acceso de seguridad local más básicos en un sistema Linux:

`/etc/passwd`
: Todos los atributos POSIX de una cuenta de usuario local del sistema, que no sean hash de contraseña, Es legibles por todos.

`/etc/group`
: Todos los atributos POSIX de la cuenta de grupo local del sistema. Es legibles por todos.

`/etc/shadow`
: Todos los hash de contraseña de usuario local del sistema (e información de caducidad). Es ilegibles por cualquier (solo procesos seleccionados).

`/etc/sudoers`
: Toda la información/asignación de privilegio local del sistema por el comando `sudo`.

Los siguientes comandos se discutieron en esta lección:

`id`
: Enumera los ID de usuario y grupo reales (o efectivas)

`last`
: Lista de usuarios que iniciaron sesión por última vez.

`who`
: Lista de usuarios que actualmente están conectados.

`w`
: Similar a `who` pero con contexto adicional.

`su`
: Cambie a otro usuario con un shell de inicio de sesión o ejecute comandos como ese usuario

pasando la contraseña del mismo.

`sudo`

Cambia el usuario (o superusuario) si tiene los permisos. El usuario actual ingresa su propia contraseña (si es necesario) para aumentar el privilegio.

`chsh`

Cambiar el shell de un usuario.

`chfn`

Cambia la información del usuario en el campo GECOS.

Respuestas a los ejercicios guiados

1. Considere la siguiente salida del comando `id`:

   ```
   $ id emma
   uid=1000(emma) gid=1000(emma)
   groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
   ```

   ¿En qué archivos se almacenan los siguientes atributos?

UID y GID	`/etc/passwd`
Groups	`/etc/group`

   - Además, ¿en qué archivo se almacena la contraseña del usuario?

     La contraseña de usuario hash se almacena en `/etc/shadow`.

2. ¿Cuál de los siguientes tipos de criptografía se usa de manera predeterminada para almacenar contraseñas localmente en un sistema Linux?

   Por defecto, se utiliza un hash unidireccional para almacenar contraseñas.

3. Si una cuenta tiene un ID de usuario (UID) enumerada bajo 1000. ¿Qué tipo de cuenta es esta?

   Las cuentas con un UID inferior a 1000 generalmente son cuentas del sistema.

4. ¿Cómo puede obtener una lista de los inicios de sesión activos en su sistema y también un recuento de ellos?

   Use el comando `w`. Además de una lista de todos los inicios de sesión activos, también mostrará información como cuántos usuarios han iniciado sesión, junto con la carga del sistema y el tiempo de actividad.

5. Usando el comando `grep`, obtuvimos el resultado siguiente con la información sobre el usuario `emma`.

   ```
   $ grep emma /etc/passwd
   emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
   ```

   Complete los espacios en blanco del gráfico con la información apropiada usando la salida del

comando anterior.

Username	`emma`
Password	`x` - should always be `x` for a valid, active user login
UID	`1000`
Primary GID	`1000`
GECOS	`Emma Smith,42 Douglas St,555.555.5555`
Home Directory	`/home/emma`
Shell	`/bin/ksh`

Respuestas a los ejercicios exploratorios

1. Compare los resultados de `last` con `w` y `who`. En comparación, ¿Qué detalles faltan a cada uno de los comandos ?

   Las herramientas `w` y `who` solo enumeran los usuarios actuales que han iniciado sesión en el sistema, mientras que `last` también enumera los usuarios que se han desconectado. El comando `w` enumera la utilización del sistema, mientras que `who` no lo hace.

2. Intente emitir los siguientes comandos `who` y `w -his`.

   - ¿Qué información se ha eliminado de la salida del comando `w` con las opciones "no header" (`-h`) y "short" (`-s`)?

     El encabezado no se imprime, lo cual es útil para el análisis, el tiempo de inicio de sesión y la información de la CPU seleccionada no se enumere.

   - ¿Qué información se ha agregado en la salida del comando `w` con la opción "ip address" (`-i`)?

     Esto imprime la dirección IP, en lugar de intentar la resolución DNS, imprime el nombre de host. Esta opción para `w` coincide mejor con la salida predeterminada del comando `last`.

     ```
     ¿Cúal archivo es el que almacena el hash de contraseña unidireccional de
     una cuenta de usuario?
     ```

     El archivo `/etc/shadow` almacena el hash de contraseña unidireccional de una cuenta de usuario, ya que no es legible por una cuenta de usuario normal y sin privilegios a diferencia del archivo `/etc/passwd`.

3. ¿Qué archivo contiene la lista de grupos de los que es miembro una cuenta de usuario? ¿Qué lógica podría usarse para compilar una lista de grupos de los que es miembro una cuenta de usuario?

   El archivo `/etc/group` tiene una lista CSV de nombres de usuario en el último campo, "members", de cualquier línea para un grupo.

   Cualquier línea en el archivo `/etc/group` donde el usuario aparece en el campo final, "members", significaría que el usuario es miembro de ese grupo, suponiendo que esté formateado correctamente (delimitado por CSV). Además, la membresía del grupo principal del usuario en el archivo `/etc/passwd` también tendrá una entrada coincidente en el archivo `/etc/group` tanto para el nombre del grupo como para el GID.

4. Por defecto uno o más (1+) de los siguientes archivos no son legibles por usuarios normales y sin privilegios. ¿Cuáles son?

   - `/etc/group`
   - `/etc/passwd`
   - `/etc/shadow`
   - `/etc/sudoers`

   Los archivos `/etc/shadow` y `/etc/sudoers` no se pueden leer de forma predeterminada, excepto por los servicios seleccionados o el superusuario. Estas respuestas serán personalizadas en función de los sistemas y nombres de usuario utilizados en el laboratorio.

5. ¿Cómo puede cambiar el shell de inicio de sesión del usuario actual al Korn Shell (`/usr/bin/ksh`) en modo no interactivo?

   ```
   $ chsh -s /usr/bin/ksh
   ```

6. ¿Por qué el directorio de inicio del usuario `root` no está ubicado dentro del directorio `/home`?

   Debido a que la cuenta `root` es necesaria para solucionar problemas y corregir errores, que pueden incluir problemas del sistema de archivos relacionados con el directorio `/home`. En tales casos, `root` debería ser completamente funcional incluso cuando el sistema de archivos `/home` aún no esté disponible.



5.2 Creating Users and Groups

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 5.2

Importancia

2

Áreas de conocimiento clave

- Comandos de usuario y de grupo
- Identificadores de los usuarios (IDs)

Lista parcial de archivos, términos y utilidades

- `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/skel/`
- `useradd`, `groupadd`
- `passwd`



5.2 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	5 Seguridad y permisos de archivos
Objetivo:	5.2 Creando usuarios y grupos
Lección:	1 de 1

Introducción

Administrar usuarios y grupos en un equipo con Linux es uno de los aspectos clave de la administración del sistema. De hecho, Linux es un sistema operativo multiusuario en el que varios de estos pueden usar la misma máquina al mismo tiempo.

La información sobre usuarios y grupos se almacena en cuatro archivos dentro del árbol de directorios `/etc/`:

`/etc/passwd`
: Es un archivo de siete campos delimitados por dos puntos que contiene información básica sobre los usuarios.

`/etc/group`
: Es un archivo de cuatro campos delimitados por dos puntos que contienen información básica sobre grupos.

`/etc/shadow`
: Es un archivo de nueve campos delimitados por dos puntos que contienen contraseñas cifradas de

los usuarios.

`/etc/gshadow`

Es un archivo de cuatro archivos delimitados por dos puntos que contienen contraseñas cifradas de los grupos.

Todos estos archivos se actualizan mediante un conjunto de herramientas de línea de comandos para la administración de usuarios y grupos, las cuales discutiremos más adelante en esta lección. También se pueden administrar mediante aplicaciones gráficas según la distribución que proporcionan interfaces más simples e inmediatas.

WARNING Aunque los archivos son texto sin formato, no los edite directamente. Utilice siempre las herramientas proporcionadas con su distribución para este propósito.

El archivo `/etc/passwd`

`/etc/passwd` es un archivo legible por todos (world-readable) que contiene una lista de usuarios en cada línea:

```
frank:x:1001:1001::/home/frank:/bin/bash
```

Cada línea constante de siete campos delimitados por dos puntos:

Username

El nombre utilizado cuando el usuario inicia sesión en el sistema.

Password

La contraseña cifrada (o una `x` si se usan contraseñas ocultas).

User ID (UID)

El número de ID asignado al usuario en el sistema.

Group ID (GID)

El número de grupo primario del usuario en el sistema.

GECOS

Un campo de comentario opcional, que se utiliza para agregar información adicional sobre el usuario (como el nombre completo). El campo puede contener múltiples entradas separadas por

comas.

Home directory

La ruta absoluta del directorio de inicio del usuario.

Shell

La ruta absoluta del programa que se inicia automáticamente cuando el usuario inicia sesión en el sistema (generalmente un shell interactivo como `/bin/bash`).

El archivo `/etc/group`

`/etc/group` es un archivo legible en todos (world-readable) que contiene una lista de grupos, cada uno en una línea separada:

```
developer:x:1002:
```

Cada línea constante de cuatro campos delimitados por dos puntos:

Group Name

El nombre del grupo.

Group Password

La contraseña cifrada del grupo (o una `x` si se usan contraseñas ocultas).

Group ID (GID)

El número de identificación asignado al grupo en el sistema.

Member list

Una lista delimitada por comas de los usuarios que pertenecen al grupo, excepto aquellos para quienes este es el grupo primario.

El archivo `/etc/shadow`

`/etc/shadow` es un archivo legible solo por usuarios root y usuarios con privilegios de root. Además contiene las contraseñas cifradas de los usuarios seperadas por una línea

```
frank:$6$i9gjM4Md4MuelZCd$7jJa8Cd2bbADFH4dwtfvTvJLOYCCCBf/.jYbK1IMYx7Wh4fErXcc2xQVU
2N1gb97yIYaiqH.jjJammzof2Jfr/:18029:0:99999:7:::
```

Cada línea consta de nueve campos delimitados por dos puntos:

Username
El nombre utilizado cuando el usuario inicia sesión en el sistema.

Encrypted password
La contraseña cifrada del usuario (si el valor es `!`, La cuenta está bloqueada).

Date of last password change
La fecha del último cambio de contraseña, como número de días desde 01/01/1970. Un valor de `0` significa que el usuario debe cambiar la contraseña en el siguiente acceso.

Minimum password age
El número mínimo de días después de un cambio de contraseña que debe pasar antes de que el usuario pueda cambiarla nuevamente.

Maximum password age
El número máximo de días que deben transcurrir antes de que se requiera un cambio de contraseña.

Password warning period
El número de días antes de que caduque la contraseña, durante los cuales se advierte al usuario que se debe cambiarla.

Password inactivity period
El número de días después de que caduca una contraseña durante el cual el usuario debe actualizarla. Después de este período, si el usuario no cambia la contraseña, la cuenta se deshabilitará.

Account expiration date
La fecha, como número de días desde el 01/01/1970 en que se deshabilitará la cuenta de usuario. Un campo vacío significa que la cuenta de usuario nunca caducará.

A reserved field
Un campo reservado para uso futuro.

El archivo `/etc/gshadow`

`/etc/gshadow` es un archivo legible solo por root y por usuarios con privilegios de root que contiene contraseñas cifradas para grupos, cada uno en una línea separada:

```
developer:$6$7QUIhUX1WdO6$H7kOYgsboLkDseFHpk04lwAtweSUQHipoxIgo83QNDxYtYwgmZTCU0qSC
uCkErmyR263rvHiLctZVDR7Ya9Ai1::
```

Cada línea constante de cuatro campos delimitados por dos puntos:

Group name

El nombre del grupo.

Encrypted password

La contraseña cifrada para el grupo (se usa cuando un usuario que no es miembro del grupo, desea unirse al grupo usando el comando `newgrp`, si la contraseña comienza con `!`. Nadie puede acceder al grupo con `newgrp`).

Group administrators

Una lista delimitada por comas de los administradores del grupo (pueden cambiar la contraseña del grupo y pueden agregar o eliminar miembros del grupo con el comando `gpasswd`).

Group members

Una lista delimitada por comas de los miembros del grupo.

Ahora que hemos visto dónde se almacena la información de usuarios y grupos, hablemos de las herramientas de línea de comandos más importantes para actualizar estos archivos.

Agregar y eliminar cuentas de usuario

En Linux, puede agregar una nueva cuenta de usuario con el comando `useradd` y puede eliminar una cuenta de usuario con el comando `userdel`.

Si desea crear una nueva cuenta de usuario llamada `frank` con una configuración predeterminada, puede ejecutar lo siguiente:

```
# useradd frank
```

Después de crear el nuevo usuario, puede establecer una contraseña usando `passwd`:


```
# passwd frank
Changing password for user frank.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Ambos comandos requieren autorización de root. Cuando ejecuta el comando `useradd`, la información del usuario, así como del grupo es almacenada y las bases de datos del grupo se actualizan para la cuenta de usuario recién creada. Si se especifica, se crea el directorio de inicio del nuevo usuario y un grupo con el mismo nombre de la cuenta.

> **TIP**
>
> Recuerde que siempre puede usar la utilidad `grep` para filtrar la contraseña y las bases de datos de grupo, mostrando solo la entrada que se refiere a un usuario o grupo específico. Para el ejemplo anterior puedes usar
>
> `cat /etc/passwd | grep frank`
>
> o
>
> `grep frank /etc/passwd`
>
> para ver información básica sobre la nueva cuenta `frank`.

Las opciones más importantes que se aplican al comando `useradd` son:

`-c`

Crea una nueva cuenta de usuario con comentarios personalizados (por ejemplo, nombre completo).

`-d`

Crea una nueva cuenta de usuario con un directorio de inicio personalizado.

`-e`

Crea una nueva cuenta de usuario estableciendo una fecha específica en la que se deshabilitará.

`-f`

Crea una nueva cuenta de usuario estableciendo el número de días después de que caduque la contraseña durante los cuales el usuario debe actualizar la contraseña.

`-g`
: Crea una nueva cuenta de usuario con un GID específico

`-G`
: Crea una nueva cuenta de usuario agregándola a múltiples grupos secundarios.

`-m`
: Crea una nueva cuenta de usuario con su directorio de inicio.

`-M`
: Crea una nueva cuenta de usuario sin su directorio de inicio.

`-s`
: Crea una nueva cuenta de usuario con un shell de inicio de sesión específico.

`-u`
: Crea una nueva cuenta de usuario con un UID específico.

Una vez que se crea la nueva cuenta de usuario, puede usar los comandos `id` y `groups` para averiguar su UID, GID y los grupos a los que pertenece.

```
# id frank
uid=1000(frank) gid=1000(frank) groups=1000(frank)
# groups frank
frank : frank
```

TIP Recuerde verificar y eventualmente editar el archivo `/etc/login.defs`, que define los parámetros de configuración que controlan la creación de usuarios y grupos. Por ejemplo, puede establecer el rango de UID y GID que se puede asignar a las nuevas cuentas de usuarios y grupos, así como especificar que no necesita usar la opción `-m` para crear el directorio de inicio del nuevo usuario y si el sistema debería crea automáticamente un nuevo grupo para cada nuevo usuario.

Si desea eliminar una cuenta de usuario, puede usar el comando `userdel`. En particular, este comando actualiza la información almacenada en las bases de datos de la cuenta, eliminando todas las entradas que se refieren al usuario especificado. La opción `-r` también elimina el directorio de inicio del usuario y todos sus contenidos, junto con la cola de correo del usuario. Otros archivos, ubicados en otro lugar, deben buscarse y eliminarse manualmente.

```
# userdel -r frank
```

Como antes, usted necesita autorización de root para eliminar cuentas de usuario.

El directorio `skel`

Cuando agrega una nueva cuenta de usuario, incluso al crear su directorio de inicio, el directorio de inicio recién creado se llena con archivos y carpetas que se copian del directorio de skel (por defecto `/etc/skel`). La idea detrás de esto, es simple: un administrador del sistema quiere agregar nuevos usuarios que tengan los mismos archivos y directorios en su hogar. Por lo tanto, si desea personalizar los archivos y carpetas que se crean automáticamente en el directorio de inicio de las nuevas cuentas de usuario, debe agregar estos nuevos archivos y carpetas al directorio de skel.

TIP Tenga en cuenta que los archivos de perfil que generalmente se encuentran en el directorio de skel son archivos ocultos. Por lo tanto, si desea enumerar todos los archivos y carpetas en el directorio skel que se copiarán en el directorio de inicio de los usuarios (recién creados), debe usar el comando `ls -Al`.

Agregar y eliminar grupos

En cuanto a la gestión de grupos, puede agregar o eliminar grupos utilizando los comandos `groupadd` y `groupdel`.

Si desea crear un nuevo grupo llamado `desarrollador`, puede ejecutar el siguiente comando como root:

```
# groupadd -g 1090 developer
```

La opción `-g` de este comando crea un grupo con un GID específico.

Si desea eliminar el grupo `desarrollador`, puede ejecutar lo siguiente:

```
# groupdel developer
```

WARNING

Recuerde que cuando agrega una nueva cuenta de usuario, el grupo primario y los grupos secundarios a los que pertenece deben existir antes de iniciar el comando `useradd`. Además, no puede eliminar un grupo si es el grupo principal de una cuenta de usuario.

El comando `passwd`

Este comando se usa principalmente para cambiar la contraseña de un usuario. Cualquier usuario puede cambiar su contraseña, pero solo el usuario root puede cambiar la contraseña de cualquier usuario.

Dependiendo de la opción `passwd` utilizada, puede controlar aspectos específicos del envejecimiento de la contraseña:

`-d`
: Elimina la contraseña de una cuenta de usuario (estableciendo así una contraseña vacía y convirtiéndola en una cuenta sin contraseña).

`-e`
: Fuerza a la cuenta de usuario a cambiar la contraseña.

`-l`
: Bloquea la cuenta de usuario (la contraseña cifrada tiene el prefijo de un signo de exclamación).

`-u`
: Desbloquea la cuenta de usuario (elimina el signo de exclamación).

`-S`
: Muestra información sobre el estado de la contraseña para una cuenta específica.

Estas opciones están disponibles solo para root. Para ver la lista completa de opciones, consulte las páginas del manual.

Ejercicios guiados

1. Para cada una de las siguientes entradas, indique el archivo al que hace referencia:

   - `developer:x:1010:frank,grace,dave`

   - `root:x:0:0:root:/root:/bin/bash`

   - `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

   - `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

   - `staff:!:dave:carol,emma`

2. Observe el siguiente resultado para responder las siguientes siete preguntas:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbcDeFgHiLmNoPqRsTu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

   - ¿Cuál es el ID de usuario (UID) y el ID de grupo (GID) de `carol`?

- ¿Qué shell está configurado para `dave` y `henry`?
- ¿Cuál es el nombre del grupo principal de `henry`?
- ¿Cuáles son los miembros del grupo `web_developer`? ¿Cuáles de estos son administradores de grupo?
- ¿Qué usuario no puede iniciar sesión en el sistema?
- ¿Qué usuario debe cambiar la contraseña la próxima vez que inicie sesión en el sistema?
- ¿Cuántos días deben pasar antes de que se requiera un cambio de contraseña para `carol`?

Ejercicios exploratorios

1. Trabajando como root, ejecute el comando `useradd -m dave` para agregar una nueva cuenta de usuario. ¿Qué operaciones realiza este comando? Suponga que `CREATE_HOME` y `USERGROUPS_ENAB` en `/etc/login.defs` están configuradas en `yes`.

2. Ahora que ha creado la cuenta `dave`, ¿puede este usuario iniciar sesión en el sistema?

3. Identifique el ID de usuario (UID) y el ID de grupo (GID) de `dave` y todos los miembros del grupo `dave`.

4. Cree los grupos `sys_admin`, `web_admin` y `db_admin` e identifique sus ID de grupo (GID).

```
Agregue una nueva cuenta de usuario llamada `carol` con UID 1035 y configure
`sys_admin` como su grupo principal y `web_admin` y `db_admin` como sus grupos
secundarios.
```

5. Elimine las cuentas de usuario `dave` y `carol` y los grupos `sys_admin`, `web_admin` y `db_admin` que haya creado previamente.

6. Ejecute el comando `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` y describa el resultado que le da en términos de permisos de archivo. ¿Cuál de estos cuatro archivos está sombreado (shadowed) por razones de seguridad? Suponga que su sistema usa contraseñas ocultas.

7. Ejecute el comando `ls -l /usr/bin/passwd`. ¿Qué bit especial se establece y cuál es su significado?

Resumen

En esta lección usted aprendió:

- Los fundamentos de la gestión de usuarios y grupos en Linux.
- Administrar información de usuarios y grupos almacenada en contraseñas y bases de datos grupales.
- Mantener el directorio skel.
- Agregar y eliminar cuentas de usuario.
- Agregar y eliminar cuentas grupales.
- Cambiar la contraseña de las cuentas de usuario.

Los siguientes comandos se discutieron en esta lección:

`useradd`
: Crear una nueva cuenta de usuario

`groupadd`
: Creae una nueva cuenta de grupo.

`userdel`
: Eliminar una cuenta de usuario.

`groupdel`
: Eliminar una cuenta grupal.

`passwd`
: Cambiar la contraseña de las cuentas de usuario y controle todos los aspectos del antigüedad de la contraseña.

Respuestas a los ejercicios guiados

1. Para cada una de las siguientes entradas, indique el archivo al que hace referencia:

   - `developer:x:1010:frank,grace,dave`

     `/etc/group`

   - `root:x:0:0:root:/root:/bin/bash`

     `/etc/passwd`

   - `henry:$1$.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

     `/etc/shadow`

   - `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

     `/etc/passwd`

   - `staff:!:dave:carol,emma`

     `/etc/gshadow`

2. Observe el siguiente resultado para responder las siguientes siete preguntas:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbcDeFgHiLmNoPqRsTu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

   - ¿Cuál es el ID de usuario (UID) y el ID de grupo (GID) de `carol`?

El UID es 1051 y el GID es 1015 (los campos tercero y cuarto en `/etc/passwd`).

- ¿Qué shell está configurado para `dave` y `henry`?

  `dave` usa /bin/bash y `henry` usa /bin/tcsh (el séptimo campo en `/etc/passwd`).

- ¿Cuál es el nombre del grupo principal de `henry`?

  El nombre del grupo es `web_admin` (el primer campo en `/etc/group`).

- ¿Cuáles son los miembros del grupo `web_developer`? ¿Cuáles de estos son administradores de grupo?

  Los miembros son `grace`, `kevin` y `christian` (el cuarto campo en `/etc/group`), pero solo `kevin` es el administrador del grupo (el tercer campo en `/etc/gshadow`).

- ¿Qué usuario no puede iniciar sesión en el sistema?

  La cuenta de usuario `henry` está bloqueada (tiene un signo de exclamación delante de hash de contraseña en `/etc/shadow`).

- ¿Qué usuario debe cambiar la contraseña la próxima vez que inicie sesión en el sistema?

  Si el tercer campo (Fecha del último cambio de contraseña) en `/etc/shadow` es 0, el usuario debe cambiar su contraseña la próxima vez que inicie sesión en el sistema. Por lo tanto, `dave` deberá cambiarla.

- ¿Cuántos días deben pasar antes de que se requiera un cambio de contraseña para `carol`?

  60 días (el quinto campo en `/etc/shadow`).

Respuestas a los ejercicios exploratorios

1. Trabajando como root, ejecute el comando `useradd -m dave` para agregar una nueva cuenta de usuario. ¿Qué operaciones realiza este comando? Suponga que `CREATE_HOME` y `USERGROUPS_ENAB` en `/etc/login.defs` están configuradas en `yes`.

   El comando agrega un nuevo usuario, llamado `dave` a la lista de usuarios en el sistema. Se crea el directorio de inicio de `dave` (por defecto `/home/dave`) y los archivos y directorios contenidos en el directorio skel se copian en el directorio de inicio. Finalmente, se crea un nuevo grupo con el mismo nombre de la cuenta de usuario.

2. Ahora que ha creado la cuenta `dave`, ¿puede este usuario iniciar sesión en el sistema?

   No, porque la cuenta `dave` está bloqueada (vea el signo de exclamación en `/etc/shadow`).

   ```
   # cat /etc/shadow | grep dave
   dave:!:18015:0:99999:7:::
   ```

   Si configura una contraseña para `dave`, la cuenta se desbloqueará. Puede hacer esto usando el comando `passwd`.

   ```
   # passwd dave
   Changing password for user dave.
   New UNIX password:
   Retype new UNIX password:
   passwd: all authentication tokens updated successfully.
   ```

3. Identifique el ID de usuario (UID) y el ID de grupo (GID) de `dave` y todos los miembros del grupo `dave`.

   ```
   # cat /etc/passwd | grep dave
   dave:x:1015:1019::/home/dave:/bin/sh
   # cat /etc/group | grep 1019
   dave:x:1019:
   ```

   El UID y el GID de `dave` son 1015 y 1019 respectivamente (los campos tercero y cuarto en `/etc/passwd`) y el grupo `dave` no tiene miembros (el cuarto campo en `/etc/group` está vacío).

4. Cree los grupos `sys_admin`, `web_admin` y `db_admin` e identifique sus ID de grupo (GID).

```
# groupadd sys_admin
# groupadd web_admin
# groupadd db_admin
# cat /etc/group | grep admin
sys_admin:x:1020:
web_admin:x:1021:
db_admin:x:1022:
```

Los GID para los grupos `sys_admin`,`web_admin` y `db_admin` son 1020, 1021 y 1022 respectivamente.

```
Agregue una nueva cuenta de usuario llamada `carol` con UID 1035 y configure
`sys_admin` como su grupo principal y `web_admin` y `db_admin` como sus grupos
secundarios.
```

```
# useradd -u 1035 -g 1020 -G web_admin,db_admin carol
# id carol
uid=1035(carol) gid=1020(sys_admin)
groups=1020(sys_admin),1021(web_admin),1022(db_admin)
```

5. Elimine las cuentas de usuario `dave` y `carol` y los grupos `sys_admin`, `web_admin` y `db_admin` que haya creado previamente.

```
# userdel -r dave
# userdel -r carol
# groupdel sys_admin
# groupdel web_admin
# groupdel db_admin
```

6. Ejecute el comando `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` y describa el resultado que le da en términos de permisos de archivo. ¿Cuál de estos cuatro archivos está sombreado por razones de seguridad? Suponga que su sistema usa contraseñas ocultas.

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root    853 mag  1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag  1 08:00 /etc/gshadow
-rw-r--r-- 1 root root   1354 mag  1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag  1 08:00 /etc/shadow
```


Los archivos `/etc/passwd` y `/etc/group` son legibles en todo el mundo y están sombreados por razones de seguridad. Cuando se utilizan contraseñas ocultas, puede ver una `x` en el segundo campo de estos archivos porque las contraseñas cifradas para usuarios y grupos se almacenan en `/etc/shadow` y `/etc/gshadow`, que solo pueden leerse root y en algunos sistemas, también por miembros que pertenecen al grupo `shadow`.

7. Ejecute el comando `ls -l /usr/bin/passwd`. ¿Qué bit especial se establece y cuál es su significado?

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

El comando `passwd` tiene el bit SUID establecido (el cuarto carácter de esta línea), lo que significa que el comando se ejecuta con los privilegios del propietario del archivo (root). Así es como los usuarios comunes pueden cambiar su contraseña.



5.3 Gestión de los permisos y la propiedad de los archivos

Referencia al objetivo del LPI

Linux Essentials version 1.6, Exam 010, Objective 5.3

Importancia

2

Áreas de conocimiento clave

- Propiedad y permisos de archivos/directorios

Lista parcial de archivos, términos y utilidades

- `ls -l`, `ls -a`
- `chmod`, `chown`



5.3 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	5 Seguridad y permisos de archivos
Objetivo:	5.3 Administrar permisos de archivos y propiedad
Lección:	1 de 1

Introducción

Al ser un sistema multiusuario, Linux necesita alguna forma de rastrear quién es dueño de cada archivo y si un usuario puede o no realizar acciones en ese en el mismo. Lo anterior es para garantizar la privacidad de los usuarios que deseen mantener la confidencialidad del contenido de sus ficheros, así como para garantizar la colaboración al hacer que ciertos archivos sean accesibles para múltiples usuarios.

Esto se hace a través de un sistema de permisos de tres niveles: cada archivo en el disco es propiedad de un usuario y un grupo de usuarios y tiene tres conjuntos de permisos: uno para su propietario, otro para el grupo propetario del archivo y otro para todos los demás. En esta lección, aprenderá cómo consultar los permisos de un archivo y cómo manipularlos.

Consultar información sobre archivos y directorios

El comando `ls` se usa para obtener una lista de los contenidos de cualquier directorio. En esta forma básica, todo lo que obtienes son los nombres de archivo:

```
$ ls
Another_Directory  picture.jpg  text.txt
```

Pero hay mucha más información disponible para cada archivo, incluyendo tipo, tamaño, propiedad y más. Para ver esta información, debe solicitar a `ls` una lista de “forma larga (long form)”, utilizando el parámetro `-l`:

```
$ ls -l
total 536
drwxrwxr-x 2 carol carol   4096 Dec 10 15:57 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Cada columna en la salida anterior tiene un significado:

- La *primera* columna de la lista muestra el tipo de archivo y los permisos.

  Por ejemplo, en `drwxrwxr-x`:

  - El primer carácter, `d` indica el tipo de archivo.
  - Los siguientes tres caracteres, `rwx` indican los permisos para el propietario del archivo, también conocido como *usuario* o `u`.
  - Los siguientes tres caracteres, `rwx` indican los permisos del *grupo* que posee el archivo, también conocido como `g`.
  - Los últimos tres caracteres, `r-x` indican los permisos para cualquier otra persona, también conocidos como *otros* u `o`.
- La *segunda* columna indica el número de enlaces duros (hard links) que apuntan a ese archivo. Para un directorio, significa el número de subdirectorios, más un enlace a sí mismo (`.`) y al directorio padre (`..`).
- Las *tercera* y *cuarta* columna muestran información de propiedad: el usuario y el grupo que posee el archivo.
- La *quinta* columna muestra el tamaño del archivo en bytes.
- La *sexta* columna muestra la fecha y hora precisas, o la marca de tiempo cuando se modificó el archivo por última vez.
- La *séptima* y última columna muestra el nombre del archivo.

Si desea ver los tamaños de archivo en formato “legible para humanos”, agregue el parámetro `-h` a al comando `ls`. Los archivos de menos de un kilobyte tendrán el tamaño que se muestra en bytes. Los archivos con más de un kilobyte y menos de un megabyte tendrán una `K` agregada después del tamaño, lo que indica que el tamaño está en kilobytes. Lo mismo sigue para los tamaños de archivo en los rangos de megabytes (`M`) y gigabytes (`G`):

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
-rw------- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

Para mostrar solo información sobre un conjunto específico de archivos, agregue los nombres de estos archivos a `ls`:

```
$ ls -lh HugeFile.zip test.sh
total 1,2G
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
---xr-xr-x 1 carol carol   33 Dec 11 10:36 test.sh
```

¿Qué pasa con los directorios?

Si intenta consultar información sobre un directorio usando `ls -l`, en su lugar le mostrará una lista del contenido del directorio:

```
$ ls -l Another_Directory/
total 0
-rw-r--r-- 1 carol carol 0 Dec 10 17:59 another_file.txt
```

Para evitar esto y consultar información sobre el directorio en sí, agregue el parámetro `-d` a `ls`:

```
$ ls -l -d Another_Directory/
drwxrwxr-x 2 carol carol 4096 Dec 10 17:59 Another_Directory/
```

Ver archivos ocultos

El listado anterior del directorio que hemos recuperado usando `ls -l` está incompleto:

```
$ ls -l
total 544
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
```

Hay otros tres archivos en ese directorio, pero están ocultos. En Linux, los archivos cuyo nombre comienza con un punto (`.`) se ocultan automáticamente. Para verlos necesitamos agregar el parámetro `-a` al comando `ls`:

```
$ ls -l -a
total 544
drwxrwxr-x 3 carol carol   4096 Dec 10 16:01 .
drwxrwxr-x 4 carol carol   4096 Dec 10 15:56 ..
drwxrwxr-x 2 carol carol   4096 Dec 10 17:59 Another_Directory
-rw------- 1 carol carol 539663 Dec 10 10:43 picture.jpg
-rw-rw-r-- 1 carol carol   1881 Dec 10 15:57 text.txt
-rw-r--r-- 1 carol carol      0 Dec 10 16:01 .thisIsHidden
```

El archivo `.thisIsHidden` simplemente está oculto porque su nombre comienza con `.`.

Sin embargo, los directorios `.` y `..` son especiales. `.` es un puntero al directorio actual, mientras que `..` es un puntero al directorio padre (el directorio que contiene el directorio actual). En Linux, cada directorio contiene al menos estos dos directorios especiales.

TIP	Puede combinar múltiples parámetros para `ls` (y muchos otros comandos de Linux). `ls -l -a`, por ejemplo, puede escribirse como `ls -la`.

Comprendiendo los tipos de archivos

Ya hemos mencionado que la primera letra en cada salida de `ls -l` describe el tipo de archivo. Los tres tipos de archivos más comunes son:

`-` (archivo normal)

Un archivo puede contener datos de cualquier tipo. Los archivos se pueden modificar, mover, copiar y eliminar.

`d` (directorio)

Un directorio contiene otros archivos o directorios y ayuda a organizar el sistema de archivos. Técnicamente, los directorios son un tipo especial de archivo.

`l` (enlace suave)

Este "archivo" es un puntero a otro archivo o directorio en otra parte del sistema de archivos.

Además de esos, hay otros tres tipos de archivos que al menos debería conocer, pero que están fuera del alcance de esta lección:

`b` (dispositivo de bloque)

Este archivo representa un dispositivo virtual o físico, generalmente discos u otros tipos de dispositivos de almacenamiento. Por ejemplo, el primer disco duro del sistema podría estar representado por `/dev/sda`.

`c` (dispositivo de caracteres)

Este archivo representa un dispositivo virtual o físico. Los terminales (como el terminal principal en `/dev/ttyS0`) y los puertos seriales son ejemplos comunes de dispositivos de caracteres.

`s` (socket)

Los sockets sirven como "conductos" para pasar información entre dos programas.

WARNING	A menos que sepa lo que está haciendo, no modifique ninguno de los permisos en dispositivos de bloque, dispositivos de caracteres o sockets. ¡Esto podría evitar que su sistema funcione incorrectamente!

Comprendiendo los permisos

En la salida de `ls -l`, los permisos de archivo se muestran justo después del tipo de archivo, como tres grupos de tres caracteres cada uno, en el orden `r`, `w` y `x`. Tenga en cuenta que un guión `-` representa la falta de un permiso particular.

Permisos en archivos

`r`

Significa *read (leer)* y tiene un valor octal de `4` (no se preocupe, hablaremos de los octales en breve). Esto significa permiso para abrir un archivo y leer su contenido.

w

Significa *write (escribir)* y tiene un valor octal de `2`. Esto significa permiso para editar o eliminar un archivo.

x

Significa *execute (ejecutar)* y tiene un valor octal de `1`. Esto significa que el archivo se puede ejecutar como un script.

Siguiendo lo explicado, un archivo con permisos `rw-` se puede leer y escribir, pero no se puede ejecutar.

Permisos en directorios

r

Significa *read* y tiene un valor octal de `4`. Esto significa permiso para leer el contenido del directorio, como los nombres de archivo, pero no implica permiso para leer los archivos ellos mismos.

w

Significa *write* y tiene un valor octal de `2`. Esto significa permiso para crear o eliminar archivos en un directorio o cambiar sus nombres, permisos y propietarios. Si un usuario tiene permiso de escritura en un directorio, el usuario puede cambiar los permisos de cualquier archivo en el directorio, incluso si el usuario no tiene permisos en el archivo o si el archivo es propiedad de otro usuario.

x

Significa *execute* y tiene un valor octal de `1`. Esto significa permiso para ingresar a un directorio, pero no para enumerar sus archivos (para eso se necesita el permiso `r`).

El último bit sobre directorios puede parecer un poco confuso. Imaginemos, que tiene un directorio llamado `Another_Directory` con los siguientes permisos:s

```
$ ls -ld Another_Directory/
d--xr-xr-x 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

También imagine que dentro de este directorio tiene un script de shell llamado `hello.sh` con los siguientes permisos:

```
-rwxr-xr-x 1 carol carol 33 Dec 20 18:46 hello.sh
```

Si usted es el usuario `carol` e intenta enumerar el contenido de `Another_Directory`, recibirá un mensaje de error, ya que su usuario no tiene permiso de lectura para ese directorio:

```
$ ls -l Another_Directory/
ls: cannot open directory 'Another_Directory/': Permission denied
```

Sin embargo, el usuario `carol` *no tiene* permisos de ejecución, lo que significa que puede ingresar al directorio. Por lo tanto, el usuario `carol` puede acceder a los archivos dentro del directorio, siempre que tenga los permisos correctos *para el archivo respectivo*. En este ejemplo, el usuario tiene permisos completos para la secuencia de comandos `hello.sh`, por lo que puede ejecutar la secuencia de comandos, incluso si no puede leer el contenido del directorio que lo contiene. Todo lo que se necesita es el nombre de archivo completo.

```
$ sh Another_Directory/hello.sh
Hello LPI World!
```

Como dijimos antes, los permisos se especifican en secuencia: primero para el propietario del archivo, luego para el grupo propietario y luego para otros usuarios. Cada vez que alguien intenta realizar una acción en el archivo, los permisos se verifican de la misma manera. Primero, el sistema verifica si el usuario actual posee el archivo, y si esto es cierto, aplica solo el primer conjunto de permisos. De lo contrario, verifica si el usuario actual pertenece al grupo propietario del archivo. En ese caso, solo aplica el segundo conjunto de permisos. En cualquier otro caso, el sistema aplicará el tercer conjunto de permisos. Esto significa que si el usuario actual es el propietario del archivo, solo los permisos del propietario son efectivos, incluso si el grupo u otros permisos son más permisivos que los permisos del propietario.

Modificando permisos de archivo

El comando `chmod` se usa para modificar los permisos de un archivo y toma al menos dos parámetros: el primero describe qué permisos cambiar, y el segundo apunta al archivo o directorio donde se realizará el cambio. Sin embargo, los permisos para cambiar se pueden describir de dos maneras diferentes o "modos".

El primero, llamado *modo simbólico (symbolic mode)*, ofrece un control detallado, que le permite agregar o revocar un solo permiso sin modificar otros en el conjunto. El otro modo *modo númerico (numeric mode)* es más fácil de recordar y más rápido de usar si desea establecer todos los valores de permiso a la vez.

Ambos modos conducirán al mismo resultado final. Por ejemplo, los comandos:

```
$ chmod ug+rw-x,o-rwx text.txt
```

y

```
$ chmod 660 text.txt
```

Producirá exactamente la misma salida, un archivo con los permisos establecidos:

```
-rw-rw---- 1 carol carol  765 Dec 20 21:25 text.txt
```

Ahora comprendamos cómo funciona cada modo.

Modo simbólico

Al describir qué permisos cambiar en modo simbólico, los primeros caracteres indican los permisos que alterará: los del usuario (`u`), del grupo (`g`), de los demás (`o`) y/o a los tres (`a`).

Luego debe decirle al comando qué hacer: puede otorgar un permiso (`+`), revocar un permiso (`-`) o establecerlo en un valor específico (`=`).

Por último, especifique a qué permiso desea afectar: leer (`r`), escribir (`w`) o ejecutar (`x`).

Por ejemplo, imagine que tenemos un archivo llamado `text.txt` con el siguiente conjunto de permisos:

```
$ ls -l text.txt
-rw-r--r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Si desea otorgar permisos de escritura a los miembros del grupo que posee el archivo, debe usar el parámetro `g+w`. Es más fácil si lo piensa de esta manera: "Para el grupo (`g`), otorgue (`+`) permisos de escritura (`w`)". Entonces, el comando sería:

```
$ chmod g+w text.txt
```

Verifiquemos el resultado con `ls`:


```
$ ls -l text.txt
-rw-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

Si desea eliminar los permisos de lectura para el propietario del mismo archivo, piense en ello como: "Para el usuario (`u`) revocar (`-`), permisos de lectura (`r`)". Entonces el parámetro es `u-r`, así:

```
$ chmod u-r text.txt
$ ls -l text.txt
--w-rw-r-- 1 carol carol 765 Dec 20 21:25 text.txt
```

¿Qué sucede si queremos establecer los permisos exactamente como `rw-` para todos? Luego piense en ello como: "Para todos (`a`), sea igual (`=`), lectura (`r`), escritura (`w`) y no ejecución (`-`)". Entonces:

```
$ chmod a=rw- text.txt
$ ls -l text.txt
-rw-rw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

Por supuesto, es posible modificar múltiples permisos al mismo tiempo. En este caso, sepárelos con una coma (`,`):

```
$ chmod u+rwx,g-x text.txt
$ ls -lh text.txt
-rwxrw-rw- 1 carol carol 765 Dec 20 21:25 text.txt
```

El ejemplo anterior se puede leer como: "Para el usuario (`u`), otorgue (`+`) permisos de lectura, escritura y ejecución (`rwx`), para el grupo (`g`) revocar (`-`), ejecutar permisos (`x`)".

Cuando se ejecuta en un directorio, `chmod` solo modifica los permisos del directorio. `chmod` tiene un modo recursivo, útil cuando desea cambiar los permisos para "todos los archivos dentro de un directorio y sus subdirectorios". Para usar esto, agregue el parámetro `-R` después del nombre del comando y antes de que los permisos cambien. Observe la siguiente manera:

```
$ chmod -R u+rwx Another_Directory/
```

Este comando se puede leer como: "Recursivamente (`-R`), para el usuario (`u`), otorgue (`+`) permisos de lectura, escritura y ejecución (`rwx`)".

WARNING Tenga cuidado y piense dos veces antes de usar el interruptor `-R`, ya que es fácil cambiar los permisos en archivos y directorios que no desea cambiar, especialmente en directorios con una gran cantidad de archivos y subdirectorios.

Numeric Mode

En *modo númerico* los permisos se especifican de manera diferente: como un valor numérico de tres dígitos en notación octal, un sistema numérico de base 8.

Cada permiso tiene un valor correspondiente, y se especifican en el siguiente orden: primero lectura (`r`), que es `4`, luego escritura (`w`), que es `2` y el último ejecución (`x`), representada por `1`. Si no hay permiso, use el valor cero (`0`). Entonces, un permiso de `rwx` sería `7` (`4+2+1`) y `r-x` sería `5` (`4+0+1`).

El primero de los tres dígitos (en conjunto) representan los permisos para el usuario (`u`), el segundo para el grupo (`g`) y el tercero para el otro (`o`). Si quisiéramos establecer los permisos para un archivo como `rw-rw----`, el valor octal sería `660`:

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Además de esto, la sintaxis en *modo numérico* es la misma que en *modo simbólico*, el primer parámetro representa los permisos que desea cambiar y el segundo apunta al archivo o directorio donde se realizará el cambio

TIP Si un valor de permiso es *odd*, ¡el archivo seguramente es ejecutable!

¿Qué sintaxis usar? El *modo númerico* se recomienda si desea cambiar los permisos a un valor específico, por ejemplo, `640` (`rw-r-----`).

El *modo simbólico* es más útil si desea voltear solo un valor específico, independientemente de los permisos actuales para el archivo. Por ejemplo, puedo agregar permisos de ejecución para el usuario usando solo `chmod u+x script.sh` sin tener en cuenta o incluso tocar los permisos actuales para el grupo y otros.

Modificando la propiedad del archivo

El comando `chown` se usa para modificar la propiedad de un archivo o directorio. La sintaxis es bastante simple:

```
chown username:groupname filename
```

Por ejemplo, verifiquemos un archivo llamado `text.txt`:

```
$ ls -l text.txt
-rw-rw---- 1 carol carol 1881 Dec 10 15:57 text.txt
```

El usuario propietario del archivo es `carol`, y el grupo también es `carol`. Ahora modifiquemos el grupo que posee el archivo a otro grupo como `students`:

```
$ chown carol:students text.txt
$ ls -l text.txt
-rw-rw---- 1 carol students 1881 Dec 10 15:57 text.txt
```

Tenga en cuenta que el usuario que posee un archivo no necesita pertenecer al grupo que lo posee. En el ejemplo anterior, el usuario `carol` no necesita ser miembro del grupo `students`. Sin embargo, ella tiene que ser miembro del grupo para transferir la propiedad del grupo del archivo a ese grupo.

El usuario o grupo puede omitirse si no desea cambiarlos. Entonces, para cambiar solo el grupo que posee un archivo, usaría `chown :students text.txt`. Para cambiar solo el usuario, el comando sería `chown carol text.txt`. Alternativamente, puede usar el comando `chgrp students text.txt` para cambiar solo el grupo.

A menos que sea el administrador del sistema (root), no puede cambiar la propiedad de un archivo propiedad de otro usuario o grupo al que no pertenece. Si intenta hacer esto, recibirá el mensaje de error `Operación no permitida`.

Consultando grupos

Antes de cambiar la propiedad de un archivo, puede ser útil saber qué grupos existen en el sistema, qué usuarios son miembros de un grupo y a qué grupos pertenece un usuario. Esas tareas se pueden realizar con dos comandos, `groups` y `groupmems`.

Para ver qué grupos existen en su sistema, simplemente escriba `groups`:

```
$ groups
carol students cdrom sudo dip plugdev lpadmin sambashare
```

Y si desea saber a qué grupos pertenece un usuario, agregue el nombre de usuario como parámetro:

```
$ groups carol
carol : carol students cdrom sudo dip plugdev lpadmin sambashare
```

Para hacer lo contrario, mostrar qué usuarios pertenecen a un grupo, use `groupmems`. El parámetro `-g` especifica el grupo, y `-l` enumerará todos sus miembros:

```
$ sudo groupmems -g cdrom -l
carol
```

TIP `groupmems` solo se puede ejecutar como root, el administrador del sistema. Si actualmente no está conectado como root, agregue `sudo` antes del comando.

Permisos especiales

Además de los permisos de lectura, escritura y ejecución para usuarios, grupos y otros, cada archivo puede tener otros tres permisos especiales que pueden alterar la forma en que funciona un directorio o cómo se ejecuta un programa. Se pueden especificar en modo simbólico o numérico, y son los siguientes:

Sticky Bit

El sticky bit, también llamado *restricted deletion flag*, tiene el valor octal `1` y en modo simbólico está representado por una `t` dentro de los permisos del otro. Esto se aplica solo a los directorios, y en Linux evita que los usuarios eliminen o cambien el nombre de un archivo en un directorio a menos que sean dueños de ese archivo o directorio.

Los directorios con el conjunto de bits fijos muestran una `t` que reemplaza la `x` en los permisos para *otros* a la hora de ver la salida de `ls -l`:

```
$ ls -ld Sample_Directory/
drwxr-xr-t 2 carol carol 4096 Dec 20 18:46 Sample_Directory/
```

En el modo numérico, los permisos especiales se especifican usando una "notación de 4 dígitos", con el primer dígito representando el permiso especial para actuar. Por ejemplo, para establecer el sticky bit (valor `1`) para el directorio `Another_Directory` en modo numérico con permisos `755`, el comando sería:

```
$ chmod 1755 Another_Directory
$ ls -ld Another_Directory
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
```

Set GID

Establecer GID, también conocido como SGID o "Set Group ID bit", tiene el valor octal `2` y en modo simbólico está representado por una `s` en los permisos de *grupo*. Esto se puede aplicar a archivos ejecutables o directorios. En los archivos ejecutables, otorgará la ejecución del archivo con privilegios del grupo. Cuando se aplica a los directorios, hará que cada archivo o directorio creado debajo herede el grupo del directorio principal.

Los archivos y directorios con bit SGID muestran una `s` que reemplaza la `x` en los permisos para el *grupo* en la salida de `ls -l`:

```
$ ls -l test.sh
-rwxr-sr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Para agregar permisos SGID a un archivo en modo simbólico, el comando sería:

```
$ chmod g+s test.sh
$ ls -l test.sh
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

El siguiente ejemplo lo hará comprender mejor los efectos de SGID en un directorio. Supongamos que tenemos un directorio llamado `Sample_Directory`, propiedad del usuario `carol` y el grupo `users`, con la siguiente estructura de permisos:

```
$ ls -ldh Sample_Directory/
drwxr-xr-x 2 carol users 4,0K Jan 18 17:06 Sample_Directory/
```

Ahora, cambiemos a este directorio y usando el comando `touch`, creemos un archivo vacío dentro de él. El resultado sería el siguiente:

```
$ cd Sample_Directory/
$ touch newfile
$ ls -lh newfile
-rw-r--r-- 1 carol carol 0 Jan 18 17:11 newfile
```

Como podemos ver, el archivo es propiedad del usuario `carol` y del grupo `carol`. Pero, si el directorio tuviera establecido el permiso SGID, el resultado sería diferente. Primero, agreguemos el bit SGID al `Sample_Directory` y verifiquemos los resultados:

```
$ sudo chmod g+s Sample_Directory/
$ ls -ldh Sample_Directory/
drwxr-sr-x 2 carol users 4,0K Jan 18 17:17 Sample_Directory/
```

La `s` en los permisos del grupo indica que el bit SGID está establecido. Ahora, cambiemos a este directorio y nuevamente creemos un archivo vacío con el comando `touch`:

```
$ cd Sample_Directory/
$ touch emptyfile
$ ls -lh emptyfile
 -rw-r--r-- 1 carol users 0 Jan 18 17:20 emptyfile
```

Como podemos ver, el grupo que posee el archivo es `users`. Esto se debe a que el bit SGID hizo que el archivo heredara el propietario del grupo de su directorio principal que es `users`.

Set UID

SUID, también conocido como "Set User ID", tiene el valor octal `4` y está representado por una `s` en los permisos *user* en modo simbólico. Solo se aplica a los archivos y su comportamiento es similar al bit SGID, pero el proceso se ejecutará con los privilegios del *usuario* que posee el archivo. Los archivos con el bit SUID muestran una `s` que reemplaza la `x` en los permisos para el usuario en la salida de `ls -l`:

```
$ ls -ld test.sh
-rwsr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

Puede combinar múltiples permisos especiales en un parámetro agregándolos juntos. Entonces, para establecer SGID (valor `2`) y SUID (valor `4`) en modo numérico para el script `test.sh` con permisos `755`, debe escribir:

```
$ chmod 6755 test.sh
```

Y el resultado sería:

```
$ ls -lh test.sh
-rwsr-sr-x 1 carol carol 66 Jan 18 17:29 test.sh
```

TIP Si su terminal admite color, y en la actualidad la mayoría de ellos lo hacen, puede ver rápidamente si estos permisos especiales se configuran al observar la salida de `ls -l`. Para sticky bit, el nombre del directorio puede mostrarse en una fuente negra con fondo azul. Lo mismo se aplica a los archivos con los bits SGID (fondo amarillo) y SUID (fondo rojo). Los colores pueden ser diferentes según la distribución de Linux y la configuración de terminal que use.

Ejercicios guiados

1. Cree un directorio llamado `emptydir` usando el comando `mkdir emptydir`. Ahora, usando `ls` enumere los permisos para el directorio `emptydir`.

2. Cree un archivo vacío llamado `emptyfile` con el comando `touch emptyfile`. Ahora, usando `chmod` con notación simbólica, agregue permisos de ejecución para el propietario del archivo `emptyfile`, y elimine los permisos de escritura y ejecución para todos los demás. Haga esto usando solo un comando `chmod`.

3. ¿Cuáles serán los permisos para un archivo llamado `text.txt` después de usar el comando `chmod 754 text.txt`?

4. Supongamos que un archivo llamado `test.sh` es un script de shell con los siguientes permisos y propiedad:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - ¿Cuáles son los permisos para la propietario del archivo?

   - Si el usuario `john` ejecuta este script, ¿bajo qué privilegios del usuario se ejecutará?

   - Usando la notación numérica, ¿cuál debería ser la sintaxis de chmod para “remover” el permiso especial otorgado a este archivo?

5. Considere este archivo:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

¿Qué tipo de archivo es `sdb1`? ¿Y quién puede escribirle?

6. Considere los siguientes 4 archivos:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Escriba los permisos correspondientes para cada archivo y directorio utilizando la notación numérica de 4 dígitos.

`Another_Directory`

`foo.bar`

`HugeFile.zip`

`Sample_Directory`

Ejercicios exploratorios

1. Pruebe esto en una terminal: Cree un archivo vacío llamado `emptyfile` con el comando `touch emptyfile`. Ahora elimine todos los permisos para el archivo con `chmod 000 emptyfile`. ¿Qué sucederá si cambia los permisos para `emptyfile` pasando solo un *valor* para `chmod` en modo numérico, como `chmod 4 emptyfile`? ¿Qué pasa si usamos dos, como `chmod 44 emptyfile`? ¿Qué podemos aprender sobre la forma en que `chmod` lee el valor numérico?

2. ¿Puede ejecutar un archivo que no tiene permisos de lectura (`--x`)? ¿Por qué o por qué no?

3. Considere los permisos para el directorio temporal en un sistema Linux `/tmp`:

```
$ ls -l /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

   Usuario, grupo y otros tienen permisos completos. Pero, ¿puede un usuario normal eliminar cualquier archivo dentro de este directorio? ¿Por qué de esto?

4. Un archivo llamado `test.sh` tiene los siguientes permisos: `-rwsr-xr-x`, lo que significa que el bit SUID está configurado. Ahora, ejecute los siguientes comandos:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

   ¿Qué hicimos? ¿Qué significa la mayúscula `S`?

5. ¿Cómo crearía un directorio llamado `Box` donde todos los archivos pertenecen automáticamente al grupo `users` y solo puede ser eliminado por el usuario que los creó?

Resumen

Como sistema multiusuario, Linux necesita una forma de rastrear quién posee y quién puede acceder a cada archivo. Esto se hace a través de un sistema de permisos de tres niveles, y en esta lección aprendimos todo sobre cómo funciona este sistema.

En esta lección, ha aprendido cómo usar `ls` para obtener información sobre los permisos de archivos, cómo controlar o cambiar quién puede crear, eliminar o modificar un archivo con `chmod`, tanto en notación *numérica* como *simbólico* y cómo cambiar la propiedad de archivos con `chown` y `chgrp`.

Los siguientes comandos se discutieron en esta lección:

`ls`
: Lista de archivos, opcionalmente incluyendo detalles como permisos.

`chmod`
: Cambiar los permisos de un archivo o directorio.

`chown`
: Cambiar el usuario propietario y/o grupo de un archivo o directorio.

`chgrp`
: Cambiar el grupo propietario de un archivo o directorio.

Respuestas a los ejercicios guiados

1. Cree un directorio llamado `emptydir` usando el comando `mkdir emptydir`. Ahora, usando `ls`, enumere los permisos para el directorio `emptydir`.

   Agregue el parámetro `-d` a `ls` para ver los atributos del archivo de un directorio, en lugar de enumerar su contenido. Por lo tanto, la respuesta es:

   ```
   ls -l -d emptydir
   ```

   Puntos de bonificación si fusionó los dos parámetros en uno, como en `ls -ld emptydir`.

2. Cree un archivo vacío llamado `emptyfile` con el comando `touch emptyfile`. Ahora, usando `chmod` en notación simbólica, agregue permisos de ejecución para el propietario del archivo `emptyfile`, y elimine los permisos de escritura y ejecución para todos los demás. Haga esto usando solo un comando `chmod`.

   Piensa en ello de esta manera:

   - “Para el usuario propietario del archivo (`u`), agregue permisos de (`+`) ejecución (`x`)”, entonces sería `u+x`.
   - “Para el grupo (`g`) y otros usuarios (`o`), elimine permisos de (`-`) escritura (`w`) y ejecución (`x`)”, entonces sería `go-wx`.

     Para combinar estos dos conjuntos de permisos, agregamos una coma entre ellos. Entonces el resultado final es:

     ```
     chmod u+x,go-wx emptyfile
     ```

3. ¿Cuáles serán los permisos de un archivo llamado `text.txt` después de usar el comando `chmod 754 text.txt`?

   Recuerde que en notación numérica cada dígito representa un conjunto de tres permisos, cada uno con un valor respectivo: *read* es `4`, *write* es `2`, *execute* es `1` y ningún permiso es `0`. Obtenemos el valor de un dígito agregando los valores correspondientes para cada permiso. `7` es `4+2+1`, o `rwx`, `5` es `4+0+1`, por lo que `r-x` y `4` solo se leen, o `r--`. Los permisos para `text.txt` serían:

   ```
   rwxr-xr--
   ```

4. Supongamos que un archivo llamado `test.sh` es un script de shell con los siguientes permisos y propiedad:

```
-rwxr-sr-x 1 carol root     33 Dec 11 10:36 test.sh
```

   - ¿Cuáles son los permisos para la propietario del archivo?

     Los permisos para el propietario (2º a 4º caracteres en la salida de `ls -l`) son `rwx`, por lo que la respuesta es: “leer, escribir y ejecutar el archivo”.

   - Si el usuario `john` ejecuta este script, ¿bajo qué privilegios del usuario se ejecutará?

     Preste atención a los permisos para el *grupo*. Son `r-s`, lo que significa que el bit SGID está establecido. El grupo propietario de este archivo es `root`, por lo que el script, incluso cuando lo inicia un usuario normal, se ejecutará con privilegios de root.

   - Usando la notación numérica, ¿cuál debería ser la sintaxis de chmod para “remover” el permiso especial otorgado a este archivo?

     Podemos “remover” los permisos especiales pasando un cuarto dígito `0` a `chmod`. Los permisos actuales son `755`, por lo que el comando debe ser `chmod 0755`.

5. Considere este archivo:

```
$ ls -l /dev/sdb1
brw-rw---- 1 root disk 8, 17 Dec 21 18:51 /dev/sdb1
```

   ¿Qué tipo de archivo es `sdb1`? ¿Y quién puede escribirle?

   El primer carácter de la salida de `ls -l` muestra el tipo de archivo. `b` es un *dispositivo de bloque*, generalmente un disco (interno o externo) conectado a la máquina. El propietario (`root`) y cualquier usuario del grupo `disk` pueden escribirle.

6. Considere los siguientes 4 archivos:

```
drwxr-xr-t 2 carol carol 4,0K Dec 20 18:46 Another_Directory
----r--r-- 1 carol carol    0 Dec 11 10:55 foo.bar
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
drwxr-sr-x 2 carol users 4,0K Jan 18 17:26 Sample_Directory
```

Escriba los permisos correspondientes para cada archivo y directorio utilizando la notación numérica de 4 dígitos.

Los permisos correspondientes, en notación numérica, son los siguientes:

`Another_Directory`

Respuesta: `1755`

`1` para sticky bit, `755` para los permisos regulares (`rwx` para el usuario,`r-x` para el grupo y otros).

`foo.bar`

Respuesta: `0044`

No hay permisos especiales (por lo que el primer dígito es `0`), no hay permisos para el usuario (`---`) y solo lee (`r-r-`) para el grupo y otros.

`HugeFile.zip`

Respuesta: `0664`

No hay permisos especiales, por lo que el primer dígito es `0`. `6` (`rw-`) para el usuario y el grupo,`4` (`r-`) para los demás.

`Sample_Directory`

Respuesta: `2755`

`2` para el bit SGID, `7` (`rwx`) para el usuario, `5` (`r-x`) para el grupo y otros.

Respuestas a los ejercicios exploratorios

1. Pruebe esto en una terminal: Cree un archivo vacío llamado `emptyfile` con el comando `touch emptyfile`. Ahora elimine todos los permisos para el archivo con `chmod 000 emptyfile`. ¿Qué sucederá si cambia los permisos para `emptyfile` pasando solo un *valor* para `chmod` en modo numérico, como `chmod 4 emptyfile`? ¿Qué pasa si usamos dos, como `chmod 44 emptyfile`? ¿Qué podemos aprender sobre la forma en que `chmod` lee el valor numérico?

   Recuerde que "establecimos a cero" los permisos para `emptyfile`. Entonces, su estado inicial sería:

   ```
   ---------- 1 carol carol    0 Dec 11 10:55 emptyfile
   ```

   Ahora, intentemos con el primer comando, `chmod 4 emptyfile`:

   ```
   $ chmod 4 emptyfile
   $ ls -l emptyfile
   -------r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Se cambiaron los permisos para otros. ¿Y si probamos dos dígitos, como `chmod 44 emptyfile`?

   ```
   $ chmod 44 emptyfile
   $ ls -l emptyfile
   ----r--r-- 1 carol carol 0 Dec 11 10:55 emptyfile
   ```

   Ahora, los permisos para *grupos* y *otros* se vieron afectados. De esto, podemos concluir que en notación numérica `chmod` lee el valor “hacia atrás”, desde el dígito menos significativo (*otros*) hasta el más significativo (*user*). Si pasa un dígito, modifica los permisos para otros. Con dos dígitos modificas *grupo* y *otros*, y con tres modificas *usuario*, *grupo* y *otros* y con cuatro dígitos modificas *usuario*, *grupo*, *otros* y los permisos especiales.

2. ¿Puede ejecutar un archivo que no tiene permisos de lectura (`--x`)? ¿Por qué o por qué no?

   Al principio, la respuesta parece obvia: si tiene permiso de ejecución, el archivo debería ejecutarse. Sin embargo, sin permiso de lectura, el sistema no puede abrir el archivo y leer su contenido para ejecutarlo. Por lo tanto, sin permiso de lectura no puede ejecutar un archivo, incluso si tiene permiso de ejecución.

3. Considere los permisos para el directorio temporal en un sistema Linux `/tmp`:

```
$ ls -l /tmp
drwxrwxrwt  19 root root  16K Dec 21 18:58 tmp
```

Usuario, grupo y otros tienen permisos completos. Pero, ¿puede un usuario normal eliminar cualquier archivo dentro de este directorio? ¿Por qué de esto?

`/tmp` es lo que llamamos un directorio *world writeable*, lo que significa que cualquier usuario puede escribir en él. Pero no queremos que un usuario modifique los archivos creados por otros, por lo que se establece el sticky bit (como lo indica la `t` en los permisos para otros). Esto significa que un usuario puede eliminar archivos en `/tmp`, pero solo si creó ese archivo.

4. Un archivo llamado `test.sh` tiene los siguientes permisos: `-rwsr-xr-x`, lo que significa que el bit SUID está configurado. Ahora, ejecute los siguientes comandos:

```
$ chmod u-x test.sh
$ ls -l test.sh
-rwSr-xr-x 1 carol carol 33 Dec 11 10:36 test.sh
```

¿Qué hicimos? ¿Qué significa la mayúscula `S`?

Eliminamos los permisos de ejecución para el usuario propietario del archivo. La `s` (o `t`) toma el lugar de la `x` en la salida de `ls -l`, por lo que el sistema necesita una forma de mostrar si el usuario tiene permisos de ejecución o no. Lo hace cambiando el caso del caracter especial.

Una `s` en minúscula en el primer grupo de permisos significa que el usuario propietario del archivo tiene permisos de ejecución y que el bit SUID está configurado. Una `S` mayúscula significa que el usuario que posee el archivo carece de permisos de ejecución (`-`) y que el bit SUID está configurado.

Lo mismo puede decirse de SGID. Una `s` en minúscula en el segundo grupo de permisos significa que el grupo propietario del archivo tiene permisos de ejecución y que el bit SGID está establecido. Una `S` mayúscula significa que el grupo que posee el archivo carece de permisos de ejecución (`-`) y que el bit SGID está establecido.

Esto también es cierto para el sticky bit, representado por la `t` en el tercer grupo de permisos. Minúscula `t` significa que el sticky bit está configurado y que otros tienen permisos de ejecución. La `T` mayúscula significa que el sticky bit está configurado y que otros no tienen permisos de ejecución.

5. ¿Cómo crearía un directorio llamado `Box` donde todos los archivos pertenecen automáticamente

al grupo `users`, y solo puede ser eliminado por el usuario que los creó?

Este es un proceso de varios pasos. El primer paso es crear el directorio:

```
$ mkdir Box
```

Queremos que cada archivo creado dentro de este directorio se asigne automáticamente al grupo `usuarios`. Podemos hacer esto configurando este grupo como el propietario del directorio, y luego configurando el bit SGID en el mismo. También debemos asegurarnos de que cualquier miembro del grupo pueda escribir en ese directorio.

Dado que no nos importa cuáles son los otros permisos y queremos "voltear" solo los bits especiales, tiene sentido usar el modo simbólico:

```
$ chown :users Box/
$ chmod g+wxs Box/
```

Tenga en cuenta que si su usuario actual no pertenece al grupo `usuarios`, deberá usar el comando `sudo` antes de los comandos anteriores para hacer el cambio como root.

Ahora, para la última parte, asegúrese de que solo el usuario que creó un archivo pueda eliminarlo. Esto se hace configurando el sticky bit (representado por una `t`) en el directorio. Recuerde que se establece en los permisos para otros (`o`).

```
$ chmod o+t Box/
```

Los permisos en el directorio `Box` deben aparecer de la siguiente manera:

```
drwxrwsr-t 2 carol users  4,0K Jan 18 19:09 Box
```

Por supuesto, puede especificar SGID y el bit adhesivo utilizando solo un comando `chmod`:

```
$ chmod g+wxs,o+t Box/
```

Puntos de bonificación si pensabas en eso.



5.4 Directorios y archivos especiales

Referencia al objetivo del LPI

Linux Essentials v1.6, Exam 010, Objective 5.4

Importancia

1

Áreas de conocimiento clave

- Usar archivos y directorios temporales
- Enlaces simbólicos

Lista parcial de archivos, términos y utilidades

- `/tmp/`, `/var/tmp/` y Sticky Bit
- `ls -d`
- `ln -s`



5.4 Lección 1

Certificación:	Linux Essentials
Versión:	1.6
Tema:	5 Seguridad y permisos de archivos
Objetivo:	5.4 Directorios y archivos especiales
Lección:	1 de 1

Introducción

En Linux, todo se trata como un archivo. Sin embargo, algunos archivos reciben un trato especial, ya sea por el lugar donde están almacenados, como los archivos temporales o la forma en que interactúan con el sistema de archivos, como los enlaces. En esta lección, aprenderemos dónde se encuentran dichos archivos, cómo funcionan y cómo administrarlos.

Archivos temporales

Los archivos temporales son archivos utilizados por los programas para almacenar datos que solo se necesitan por un corto tiempo. Estos pueden ser los datos de procesos en ejecución, registros de bloqueo, archivos de memoria virtual de un autoguardado, archivos intermedios utilizados durante una conversión de archivos, archivos de caché, etc.

Ubicación de los archivos temporales

La versión 3.0 del *Filesystem Hierarchy Standard* (FHS) define las ubicaciones estándar para los archivos temporales en los sistemas Linux. Cada ubicación tiene un propósito y comportamiento diferente, y se recomienda que los desarrolladores sigan las convenciones establecidas por el FHS cuando escriban datos temporales en el disco.

`/tmp`

Según el FHS, los programas no deben asumir que los archivos escritos se conservarán entre las invocaciones de un programa. La *recomendación* es que este directorio sea limpiado (todos los archivos borrados) durante el arranque del sistema, aunque esto es *no es obligatorio*.

`/var/tmp`

Otra ubicación para los archivos temporales, pero ésta *no debe ser borrada* durante el arranque del sistema, es decir, los archivos almacenados aquí normalmente persistirán entre los reinicios.

`/run`

Este directorio contiene datos variables en tiempo de ejecución utilizados por los procesos en ejecución, como los archivos de identificación de los procesos (`.pid`). Los programas que necesitan más de un archivo de tiempo de ejecución pueden crear subdirectorios aquí. Esta ubicación *debe ser despejada* durante el arranque del sistema. El propósito de este directorio fue una vez servido por `/var/run` y en algunos sistemas `/var/run` puede ser un enlace simbólico a `/run`.

Tenga en cuenta que no hay nada que impida a un programa crear archivos temporales en otra parte del sistema, pero es una buena práctica respetar las convenciones establecidas por FHS.

Permisos en los archivos temporales

Disponer de directorios temporales en todo el sistema en un sistema multiusuario, presenta algunos problemas en lo que respecta a los permisos de acceso. Al principio se pensó que tales directorios serían "escritos por todo el mundo", es decir, que cualquier usuario podría escribir o borrar datos en ellos. Pero si esto fuera cierto, ¿cómo podríamos evitar que un usuario borre o modifique los archivos creados por otro?

La solución es un permiso especial llamado *sticky bit*, que se aplica tanto a directorios como a archivos. Sin embargo, por razones de seguridad, el kernel de Linux ignora el bit fijo cuando se aplica a los archivos. Cuando este bit especial se establece para un directorio, evita que los usuarios eliminen o renombren un archivo dentro de ese directorio a menos que sean propietarios del archivo.

Los directorios con el sicky bit muestran una `t` reemplazando la `x` en el permiso para *otros* en la salida de `ls -l`. Por ejemplo, revisemos los permisos de los directorios `tmp` y `var/tmp`:

```
$ ls -ldh /tmp/ /var/tmp/
drwxrwxrwt 25 root root 4,0K Jun  7 18:52 /tmp/
drwxrwxrwt 16 root root 4,0K Jun  7 09:15 /var/tmp/
```

Como puedes ver por la `t` que reemplaza a la `x` en el permiso para *otros*, ambos directorios tienen el sticky bit.

Para poner el sticky bit en un directorio usando `chmod` en modo numérico, usa la notación de cuatro dígitos y `1` como primer dígito. Por ejemplo:

```
$ chmod 1755 temp
```

Pondrá el sticky bit para el directorio llamado `temp` y los permisos como `rwxr-xr-t`.

Cuando use el modo simbólico, use el parámetro `t`. Entonces, `+t` para establecer el sticky bit, y `-t` para desactivarlo. Por ejemplo:

```
$ chmod +t temp
```

Comprendiendo los enlaces

Ya hemos dicho que en Linux, todo se trata como un archivo. Pero hay un tipo de archivo *especial* llamado *link*, y hay dos tipos de enlaces en un sistema Linux:

Enlaces simbólicos

También llamados *enlaces suaves* (*soft links*), apuntan a la ruta de otro archivo. Si borras el archivo al que apunta el enlace (llamado *target*) el enlace seguirá existiendo, pero “dejará de funcionar”, ya que ahora apunta a “nada”.

Enlaces duros

Piensa en un enlace duro como un segundo nombre para el archivo original. *No* son duplicados, sino una entrada adicional en el sistema de archivos que apunta al mismo lugar (inodo) en el disco.

TIP

Un *inodo* es una estructura de datos que almacena atributos para un objeto (como un archivo o directorio) en un sistema de archivos. Entre esos atributos están el nombre del archivo, los permisos, la propiedad y en qué bloques del disco se almacenan los datos para el objeto. Piensa en ello como una entrada en un índice, de ahí el nombre, que viene de “nodo de índice”.

Trabajando con enlaces duros (Hard Links)

Creación de enlaces duros

El comando para crear un enlace duro en Linux es `ln`. La sintaxis básica es:

```
$ ln TARGET LINK_NAME
```

El `TARGET` debe existir (este es el archivo al que apuntará el enlace), y si el destino no está en el directorio actual, o si quieres crear el enlace en otro lugar, *debes* definir la ruta completa. Por ejemplo, el comando:

```
$ ln target.txt /home/carol/Documents/hardlink
```

Creará un archivo llamado `hardlink` en el directorio `/home/carol/Documents/`, enlazado al archivo `target.txt` en el directorio actual.

Si se omite el último parámetro (`LINK_NAME`), se creará un enlace con el mismo nombre del objetivo en el directorio actual.

Administrando los enlaces duros

Los enlaces duros son entradas en el sistema de archivos que tienen nombres diferentes, pero que apuntan a los mismos datos en el disco. Todos esos nombres son iguales y pueden ser usados para referirse a un archivo. Si se cambia el contenido de uno de los nombres, el contenido de todos los demás nombres que apuntan a ese archivo cambia, ya que todos estos nombres apuntan a los mismos datos. Si elimina uno de los nombres, los otros nombres seguirán funcionando.

Esto sucede porque cuando "borras" un archivo, los datos no se borran realmente del disco. El sistema simplemente borra la entrada en la tabla del sistema de archivos que apunta al inodo correspondiente a los datos del disco. Pero si tienes una segunda entrada que apunta al mismo inodo, todavía puedes llegar a los datos. Piensa en ello como dos caminos que convergen en el mismo punto. Incluso si bloqueas o redireccionas uno de los caminos, todavía puedes llegar al destino usando el otro.

Puedes comprobarlo usando el parámetro `-i` de `ls`. Considere los siguientes contenidos de un directorio:


```
$ ls -li
total 224
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 hardlink
3806696 -r--r--r-- 2 carol carol 111702 Jun  7 10:13 target.txt
```

El número que precede a los permisos es el número inode. ¿Ves que tanto el archivo `hardlink` como el archivo `target.txt` tienen el mismo número (`3806696`)? Esto es porque uno es un enlace duro del otro.

Pero, ¿cuál es el original y cuál es el enlace? No se puede decir realmente, ya que para todos los propósitos prácticos son los mismos.

Los archivos temporales son archivos utilizados por los programas para almacenar datos que solo se necesitan por un corto tiempo. Estos pueden ser los datos de procesos en ejecución, registros de bloqueo, archivos de memoria virtual de un autoguardado, archivos intermedios utilizados durante una conversión de archivos, archivos de caché, etc.

A diferencia de los enlaces simbólicos, solo puede crear enlaces duros a los archivos, y tanto el enlace como el destino deben residir en el mismo sistema de archivos.

Mover y eliminar enlaces duros

Dado que los enlaces duros se tratan como archivos normales, pueden eliminarse con `rm` y cambiarse de nombre o moverse por el sistema de archivos con `mv`. Y dado que un enlace duro apunta al mismo inodo de destino (target), puede moverse libremente, sin temor a “romper” el enlace.

Enlaces simbólicos

Creando enlaces simbólicos

El comando utilizado para crear un enlace simbólico también es `ln`, pero agregando el parámetro `-s`. Al igual que:

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Esto creará un archivo llamado `softlink` en el directorio `/home/carol/Documents/`, apuntando al archivo `target.txt` en el directorio actual.

Al igual que con los enlaces duros, puede omitir el nombre del enlace para crear un enlace con el mismo nombre que el objetivo en el directorio actual.

Administrando los enlaces simbólicos

Los enlaces simbólicos apuntan a otra ruta en el sistema de archivos. Puede crear enlaces simbólicos a los archivos *y* directorios, incluso en diferentes particiones. Es bastante fácil detectar un enlace simbólico en la salida de `ls`:

```
$ ls -lh
total 112K
-rw-r--r-- 1 carol carol 110K Jun  7 10:13 target.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 10:14 softlink -> target.txt
```

En el ejemplo anterior, el primer carácter en los permisos del archivo `softlink` es `l`, indicando un enlace simbólico. Además, justo después del nombre del archivo vemos el nombre del objetivo al que apunta el enlace, el archivo `target.txt`.

Observe que en los listados de archivos y directorios, los enlaces suaves, siempre muestran los permisos `rwx` para el usuario, el grupo y otros, pero en la práctica los permisos de acceso para ellos son los mismos que los del objetivo (target).

Mover y eliminar los enlaces simbólicos

Como los enlaces duros, los enlaces simbólicos pueden ser eliminados usando `rm` y movidos o renombrados usando `mv`. Sin embargo, se debe tener especial cuidado al crearlos, para evitar “romper” el enlace si se mueve de su ubicación original.

Cuando se crean enlaces simbólicos hay que tener en cuenta que, a menos que se especifique completamente un camino, la ubicación del objetivo se interpreta como *relativa* a la ubicación del enlace. Esto puede crear problemas si el enlace, o el archivo al que apunta, se mueve.

Esto es más fácil de entender con un ejemplo. Digamos que tenemos un archivo llamado `original.txt` en el directorio actual, y queremos crear un enlace simbólico a él llamado `softlink`. Podríamos usar:

```
$ ln -s original.txt softlink
```

Y aparentemente todo estaría bien. Comprobemos con `ls`:


```
$ ls -lh
total 112K
-r--r--r-- 1 carol carol 110K Jun  7 10:13 original.txt
lrwxrwxrwx 1 carol carol   12 Jun  7 19:23 softlink -> original.txt
```

Mira cómo se construye el enlace: `softlink` apunta a (→) `original.txt`. Sin embargo, veamos qué sucede si movemos el enlace al directorio padre e intentamos mostrar su contenido usando el comando `less`:

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

Como no se especificó la ruta de acceso al `original.txt`, el sistema asume que está en el mismo directorio que el enlace. Cuando esto deja de ser cierto, el enlace deja de funcionar.

La forma de evitarlo es especificar siempre el camino completo al objetivo al crear el enlace:

```
$ ln -s /home/carol/Documents/original.txt softlink
```

De esta manera, no importa dónde muevas el enlace, este seguirá funcionando, porque apunta a la ubicación absoluta del objetivo (target). Compruébalo con `ls`:

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink ->
/home/carol/Documents/original.txt
```

Ejercicios guiados

1. Imagina que un programa necesita crear un archivo temporal de un solo uso que no se volverá a necesitar después de que el programa se cierre. ¿Cuál sería el directorio correcto para crear este archivo?

2. ¿Cuál es el directorio temporal que *debe* ser limpiado durante el proceso de arranque?

3. ¿Cuál es el parámetro para `chmod` en modo *simbólico* para habilitar el sticky bit en un directorio?

4. Imagina que hay un archivo llamado `documento.txt` en el directorio `/home/carol/Documents`. ¿Cuál es el comando para crear un enlace simbólico a hacia al archivo llamado `text.txt` en el directorio actual?

5. Explique la diferencia entre un enlace duro a un archivo y una copia de este archivo.

Ejercicios exploratorios

1. Imagina que dentro de un directorio creas un archivo llamado `recipes.txt`. Dentro de este directorio, también crearás un enlace duro a este archivo llamado `receitas.txt` y un enlace simbólico (o *soft*) a este llamado `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s recipes.txt rezepte.txt
   ```

   El contenido del directorio debería aparecer así:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 24 10:12 rezepte.txt -> receitas.txt
   ```

   Recuerde que, como un enlace duro, `receitas.txt` apunta al mismo inodo que `recipes.txt`. ¿Qué pasaría con el enlace suave `rezepte.txt` si se borra el nombre `receitas.txt`? ¿Por qué?

2. Imagina que tienes una unidad flash conectada en tu sistema, y montada en `/media/tuusuario/FlashA`. Quieres crear en tu directorio principal un enlace llamado `schematics.pdf`, apuntando al archivo `esquema.pdf` en el directorio raíz del pendrive. Entonces, escribes el comando:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   ¿Qué pasaría? ¿Por qué?

3. Considere la siguiente salida de `ls -lah`:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

- ¿Cuántos enlaces apuntan al archivo `document.txt`?

- ¿Son enlaces suaves o duros?

- ¿Qué parámetro debes pasar a `ls` para ver qué inodo ocupa cada archivo?

4. Imagina que tienes en tu directorio `~/Documentos` un archivo llamado `clients.txt` que contiene algunos nombres de clientes y un directorio llamado `somedir`. Dentro de esto hay un archivo *diferente* llamado también `clients.txt` con diferentes nombres. Para replicar esta estructura, usa los siguientes comandos.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Luego creas un enlace dentro de `somedir` llamado `partners.txt` apuntando a este archivo, con los comandos:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Entonces, la estructura del directorio es:


```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Ahora, mueva `partners.txt` de `somedir` a `~/Documentos` y enumere su contenido.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

¿Seguirá funcionando el enlace? Si es así, ¿qué archivo tendrá su contenido listado? ¿Por qué?

5. Considere los siguientes archivos:

```
-rw-r--r-- 1 carol carol 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 carol carol 11 Jun 24 11:13 partners.txt -> clients.txt
```

¿Cuáles son los permisos de acceso para "partners.txt"? ¿Por qué?

Resumen

En esta lección usted aprendió:

- ¿Donde se almacenan los archivos temporales?
- ¿Cuál es el permiso especial que se les aplica?
- ¿Qué son los enlaces?
- La diferencia entre los enlaces simbólicos y los duros.
- ¿Cómo crear enlaces?
- ¿Cómo moverlos, renombrarlos o eliminarlos?

Los siguientes comandos se discutieron en esta lección:

- `ln`
- El parámetro `-i` a `ls`

Respuestas a los ejercicios guiados

1. Imagina que un programa necesita crear un archivo temporal de un solo uso que no se volverá a necesitar después de que el programa se cierre. ¿Cuál sería el directorio correcto para crear este archivo?

   Como no nos importa el archivo después de que el programa termine de ejecutarse, el directorio correcto es `/tmp`.

2. ¿Cuál es el directorio temporal que *debe* ser limpiado durante el proceso de arranque?

   El directorio es `run` o en algunos sistemas, `/var/run`.

3. ¿Cuál es el parámetro para `chmod` en modo *simbólico* para habilitar el sticky bit en un directorio?

   El símbolo para el sticky bit en modo simbólico es `t`. Ya que queremos habilitar (agregar) este permiso al directorio, el parámetro debe ser `+t`.

4. Imagina que hay un archivo llamado `documento.txt` en el directorio `/home/carol/Documents`. ¿Cuál es el comando para crear un enlace simbólico a hacia al archivo llamado `text.txt` en el directorio actual?

   `ln -s` es el comando para crear un vínculo simbólico. Ya que debes especificar la ruta completa del archivo al que estás enlazando, el comando es:

   ```
   $ ln -s /home/carol/Documents/document.txt text.txt
   ```

5. Explique la diferencia entre un enlace duro a un archivo y una copia de este archivo.

   Un enlace duro es sólo otro nombre para un archivo. Aunque parezca un duplicado del archivo original, a todos los efectos tanto el enlace como el original son iguales, ya que apuntan a los mismos datos en el disco. Los cambios realizados en el contenido del enlace se reflejarán en el original y viceversa. Una copia es una entidad completamente independiente, que ocupa un lugar diferente en el disco. Los cambios realizados en la copia no se reflejarán en el original y viceversa.

Respuestas a los ejercicios exploratorios

1. Imagina que dentro de un directorio creas un archivo llamado `recipes.txt`. Dentro de este directorio, también crearás un enlace duro a este archivo llamado `receitas.txt` y un enlace simbólico (o *soft*) a este llamado `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s recipes.txt rezepte.txt
   ```

   El contenido del directorio debería aparecer así:

   ```
   $ ls -lhi
   total 160K
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 receitas.txt
   5388833 -rw-r--r-- 4 carol carol 77K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol  12 jun 24 10:12 rezepte.txt -> recipes.txt
   ```

   Recuerde que, como un enlace duro, `receitas.txt` apunta al mismo inodo que `recipes.txt`. ¿Qué pasaría con el enlace suave `rezepte.txt` si se borra el nombre `receitas.txt`? ¿Por qué?

   El enlace blando `rezepte.txt` dejaría de funcionar. Esto se debe a que los enlaces suaves (soft links) apuntan a nombres, no a inodos, y el nombre `receitas.txt` ya no existe, incluso si los datos siguen en el disco bajo el nombre `recetas.txt`.

2. Imagina que tienes una unidad flash conectada en tu sistema, y montada en `/media/tuusuario/FlashA`. Quieres crear en tu directorio principal un enlace llamado `schematics.pdf`, apuntando al archivo `esquema.pdf` en el directorio raíz del pendrive. Entonces, escribes el comando:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   ¿Qué pasaría? ¿Por qué?

   El comando fallaría. El mensaje de error sería `Invalid cross-device link`, y deja clara la razón: los enlaces duros no pueden apuntar a un objetivo (target) en una partición o dispositivo diferente. La única manera de crear un enlace como este es usar un enlace *simbólico* o *soft*, añadiendo el parámetro `s` a `ln`.

3. Considere la siguiente salida de `ls -lah`:

```
$ ls -lah
total 3,1M
drwxr-xr-x 2 carol carol 4,0K jun 17 17:27 .
drwxr-xr-x 5 carol carol 4,0K jun 17 17:29 ..
-rw-rw-r-- 1 carol carol 2,8M jun 17 15:45 compressed.zip
-rw-r--r-- 4 carol carol  77K jun 17 17:25 document.txt
-rw-rw-r-- 1 carol carol 216K jun 17 17:25 image.png
-rw-r--r-- 4 carol carol  77K jun 17 17:25 text.txt
```

   - ¿Cuántos enlaces apuntan al archivo `document.txt`?

     Cada archivo comienza con un recuento de enlaces de `1`. Dado que el recuento de enlaces para el archivo es de `4`, hay tres enlaces que apuntan a ese archivo.

   - ¿Son enlaces suaves o duros?

     Son enlaces duros, ya que los enlaces suaves no aumentan el número de enlaces de un archivo.

   - ¿Qué parámetro debes pasar a `ls` para ver qué inodo ocupa cada archivo?

     El parámetro es `-i`. El inode se mostrará como la primera columna en la salida de `ls`, como se ve observa a continuación:

```
$ ls -lahi
total 3,1M
5388773 drwxr-xr-x 2 rigues rigues 4,0K jun 17 17:27 .
5245554 drwxr-xr-x 5 rigues rigues 4,0K jun 17 17:29 ..
5388840 -rw-rw-r-- 1 rigues rigues 2,8M jun 17 15:45 compressed.zip
5388833 -rw-r--r-- 4 rigues rigues  77K jun 17 17:25 document.txt
5388837 -rw-rw-r-- 1 rigues rigues 216K jun 17 17:25 image.png
5388833 -rw-r--r-- 4 rigues rigues  77K jun 17 17:25 text.txt
```

4. Imagina que tienes en tu directorio `~/Documentos` un archivo llamado `clients.txt` que contiene algunos nombres de clientes y un directorio llamado `somedir`. Dentro de esto hay un archivo *diferente* llamado también `clients.txt` con diferentes nombres. Para replicar esta estructura, usa los siguientes comandos.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Luego creas un enlace dentro de `somedir` llamado `partners.txt` apuntando a este archivo, con los comandos:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Entonces, la estructura del directorio es:

```
Documents
|-- clients.txt
`-- somedir
    |-- clients.txt
    `-- partners.txt -> clients.txt
```

Ahora, mueva `partners.txt` de `somedir` a `~/Documentos` y enumere su contenido.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

¿Seguirá funcionando el enlace? Si es así, ¿qué archivo tendrá su contenido listado? ¿Por qué?

Este es un “truco”, pero el enlace funcionará, y el archivo listado será el de `Documents`, que contiene los nombres `John`, `Michael`, `Bob`.

Recuerde que como no especificó la ruta completa al destino `clients.txt` al crear el enlace blando `partners.txt`, la ubicación del destino se interpretará como relativa a la ubicación del enlace, que en este caso es el directorio actual.

Cuando el enlace fue movido de `~/Documentos/somedir` a `~/Documents`, debería dejar de funcionar, ya que el objetivo ya no estaba en el mismo directorio que el enlace. Sin embargo, sucede que hay un archivo llamado `clients.txt` en `~/Documentos`, por lo que el enlace apuntará a este archivo, en lugar del destino original dentro de `~/somedir`.

Para evitar esto, siempre especifica el camino completo al objetivo al crear un enlace simbólico.

5. Considere los siguientes archivos:

```
-rw-r--r-- 1 rigues rigues 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 rigues rigues 11 Jun 24 11:13 partners.txt -> clients.txt
```

¿Cuáles son los permisos de acceso para `partners.txt`? ¿Por qué?

Los permisos de acceso para `partners.txt` son `rw-r-r--`, ya que los enlaces siempre heredan los mismos permisos de acceso que el objetivo.

Pie de imprenta

LPI Learning Materials son una iniciativa del Linux Professional Institute (https://lpi.org). Los materiales y sus traducciones pueden encontrarse en https://learning.lpi.org.

Para preguntas y comentarios sobre esta edición, así como sobre todo el proyecto, escriba un correo electrónico a: learning@lpi.org.